CORE TASK ASSISTANCE IN VIDEO GAMES

By

Jawad Jandali Refai

B.Sc. Computer Science, University of Wollongong, 2016

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Computer Science

In the Graduate Academic Unit of Computer Science

Supervisors: Scott Bateman, PhD, Computer Science
Michael W. Fleming, PhD, Computer Science

Examing Board: Eric Aubanel, PhD, Computer Science, Chair
Weichang Du, PhD, Computer Science
Erik Scheme, PhD, Electrical and Computer Engineering

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

October, 2018

© Jawad Jandali Refai, 2019
Video games can be challenging, which is part of what makes games stimulating and entertaining. However, if they are too challenging, the player may find it frustrating. Game designers may balance their game by providing players with assistance. Previous work explores the effectiveness of potential assistance techniques within a particular genre and platform. Complex games could require several types of assistance to support a wide variety of gameplay mechanics. Designers would need to gather information from scattered sources to make informed decisions to apply optimal assistance. In this thesis, we propose a generalized framework for assistance in games, irrespective of genre or target platform. We achieve this by discussing techniques targeted at the 10 fundamental core tasks in video games that are the base of any game mechanic, such as Aiming, Reaction Time, and Visual Search. We also explore the best practices for choosing, interpreting, and implementing one of the 35 assistance techniques.
## Table of Contents

ABSTRACT.................................................................................................................................................. ii
Table of Contents......................................................................................................................................... iii
List of Tables ............................................................................................................................................... vii
List of Figures .............................................................................................................................................. viii
1 - Introduction ........................................................................................................................................... 1
   Organization........................................................................................................................................... 4
   Chapter 2 - Related Work ....................................................................................................................... 5
   Chapter 3 - Core Tasks and Assistance .................................................................................................. 5
   Chapter 4 - Assistance Techniques ......................................................................................................... 5
   Chapters 5 and 6 - Demonstration and Evaluation ............................................................................... 5
   Chapters 7, 8, and 9 - Discussions, Future Work, and Conclusion ....................................................... 5
2 - Background and Related Work ........................................................................................................... 6
   2.1 - Difficulty ....................................................................................................................................... 6
   2.2 - Flow ............................................................................................................................................... 7
   2.3 - Assistance ..................................................................................................................................... 8
3 - Core Tasks and Assistance ................................................................................................................... 9
   3.1 - Core Tasks .................................................................................................................................... 9
   3.2 - Assistance Techniques .................................................................................................................. 11
      3.2.1 - Definition ............................................................................................................................... 11
      3.2.2 - Why explore Assistance in terms of Core Tasks? ............................................................... 12
      3.2.3 - Difference between Assistance and Difficulty Adjustment ............................................. 13
      3.2.4 - Difference between Assistance and “Doing it for them” .................................................... 14
      3.2.5 - Strength of Assistance Techniques ..................................................................................... 15
      3.2.6 - Modes of Delivery .................................................................................................................. 16
      3.2.7 - Organization and Structure for Assistance Techniques ...................................................... 18
      3.2.8 - Methodology .......................................................................................................................... 18
4 - Assistance Techniques ......................................................................................................................... 20
   Core Tasks & Assistance List Template .................................................................................................. 21
4.X - Core Task ........................................................................................................................................... 21
   4.X.1 - Introduction ................................................................................................................................. 21
   4.X.2 - Examples .................................................................................................................................... 21
      4.X.2.1 Examples of Core Task Assistance ....................................................................................... 21
      4.X.2.2 Examples of Player Exploits ................................................................................................. 21
   4.X.3 - Assistance Techniques ............................................................................................................... 22
      4.X.3.N – Assistance Technique N ................................................................................................... 22
4.1 - Signal Detection ................................................................................................................................. 23
   4.1.1 – Introduction ................................................................................................................................. 23
   4.1.2 – Examples ................................................................................................................................. 23
      4.1.2.1 Examples of Signal Detection Assistance ........................................................................... 25
      4.1.2.2 Examples of Player Exploits ............................................................................................... 26
   4.1.3 – Assistance Techniques ............................................................................................................... 26
      4.1.3.1 – Companion Signals ........................................................................................................... 27
      4.1.3.2 – Augmented Parameters ................................................................................................. 29
4.1.3.3 – Announcements, Explanation, and Emphasis ........................................ 31
4.1.3.4 – Signal Priority ..................................................................................... 33
4.1.3.5 – Signal Transformation / Replacement ............................................... 36
4.1.3.6 – Noise Reduction ................................................................................. 38
4.2 – Signal Discrimination ............................................................................... 40
4.2.1 – Introduction ........................................................................................... 40
4.2.2 – Examples ............................................................................................... 40
  4.2.2.1 Examples of Signal Discrimination Assistance ...................................... 40
4.2.3 – Assistance Techniques ......................................................................... 41
  4.2.3.1 – Additional Cues ............................................................................... 41
  4.2.3.2 – Augmented Parameters ................................................................... 43
  4.2.3.3 – High Contrast .................................................................................. 45
  4.2.3.4 – Noise Reduction ............................................................................. 46
4.3 – Body Controls .......................................................................................... 48
  4.3.1 – Introduction ......................................................................................... 48
  4.3.2 – Examples .............................................................................................. 49
4.3.3 – Assistance Techniques ......................................................................... 49
  4.3.3.1 – Input Modulation (Muscle Activation) ............................................. 49
  4.3.3.2 – Detection Threshold (Body Movement) .......................................... 50
4.4 – Reaction Time .......................................................................................... 52
  4.4.1 – Introduction ......................................................................................... 52
  4.4.2 – Examples .............................................................................................. 52
  4.4.2.1 Examples of Reaction Time Assistance ............................................. 53
  4.4.3 – Assistance Techniques ....................................................................... 53
  4.4.3.1 – Additional Cues ............................................................................... 54
  4.4.3.2 – Advance Warning .......................................................................... 55
  4.4.3.3 – Reaction Window .......................................................................... 56
4.5 – Visual Search ........................................................................................... 58
  4.5.1 – Introduction ......................................................................................... 58
  4.5.2 – Examples .............................................................................................. 58
  4.5.2.1 Examples of Visual Search Assistance ............................................. 59
  4.5.2.2 Examples of Player Exploits ............................................................... 60
  4.5.3 – Assistance Techniques ....................................................................... 60
   4.5.3.1 – Highlights....................................................................................... 60
   4.5.3.2 – Visual Connection.......................................................................... 62
   4.5.3.3 – Target Details.................................................................................. 63
   4.5.3.4 – Compass ....................................................................................... 64
   4.5.3.5 – Directional Cues .......................................................................... 65
4.6 – Pointing ..................................................................................................... 67
  4.6.1 – Introduction ......................................................................................... 67
  4.6.2 – Examples .............................................................................................. 68
  4.6.2.1 Examples of Pointing Assistance ...................................................... 68
  4.6.2.2 Examples of Player Exploits ............................................................... 69
4.6.3 – Assistance Techniques ......................................................................... 69
  4.6.3.1 – Sticky Targeting ............................................................................. 70
  4.6.3.2 – Target Gravity ................................................................................ 72
4.11 - Choosing Techniques & Calibration ......................................................... 100
Step 1: Core Task Identification ................................................................. 100
Step 2: Assistance Goals .......................................................................... 100
Step 3: Selecting Techniques ................................................................... 101
Step 4: Calibration .................................................................................... 103

5 – Demonstration and Evaluation ............................................................. 104
5.1 – Demonstration .................................................................................. 104
5.2 – Evaluation ........................................................................................ 107
5.2.1 - Apparatus and Setup ...................................................................... 107
5.2.2 - Participants ................................................................................... 108
5.2.3 - Experimental Task .......................................................................... 108
5.2.4 - Techniques .................................................................................................................. 109
5.2.5 - Procedure .................................................................................................................. 111
5.2.6 - Results ...................................................................................................................... 113
5.2.7 - Summary of Results ................................................................................................. 123
6 - Discussion ......................................................................................................................... 124
6.1 - Explanation of Results ................................................................................................. 124
  6.1.1 - Why does Third-Person have a higher mean time and lower mean score than Top-Down? ........................................................................................................ 125
  6.1.2 – Did some techniques not perform differently in the Third-Person game?.. 125
  6.1.3 - Why did Highlights lead to the highest scores and lowest times?........... 126
  6.1.4 - Why was Compass the second best in terms of performance?.............. 127
  6.1.5 - Why was Hint no better than No Assistance? ................................... 128
  6.1.6 - Why did participants prefer Compass almost as much as Highlights?....... 128
  6.2 - “Highlight” Implementation Differences ................................................................. 130
  6.3 - Two-Way Performance Adjustment: “Negative Assistance” ................................. 131
  6.4 - How do the study results inform game designers about incorporating assistance into their games? ........................................................................................................ 132
7 - Conclusion and Future Work ............................................................................................ 133
  7.1 - Conclusion .................................................................................................................. 133
  7.2 - Future Work .............................................................................................................. 134
    7.2.1 - Refine List of Core Tasks ............................................................................... 134
    7.2.2 - Expand List of Assistance Techniques ............................................................. 135
    7.2.3 - Further exploration and analysis of assistance techniques .............................. 137
References ................................................................................................................................. 139
Referenced Media ................................................................................................................... 144
Ludography ............................................................................................................................. 146
Appendix A - Referenced Games ......................................................................................... 149
Appendix B - Tables and Questionnaires .......................................................................... 151
Appendix C - Curriculum Vitae ............................................................................................ 153
List of Tables

Table B-1 Non-exhaustive list of video game genres with examples of popular games for each one. The core tasks listed for each one are based on player tasks/actions found in their respective examples. ................................................................. 151

Table B-2 A short snippet of Overall test data collection. These particular entries are not from the experiment results. ................................................................. 152

Table B-3 A short snippet of Per-Click test data collection. These particular entries are not from the experiment results. ................................................................. 152
List of Figures

Figure 3.1 Extracting Core Tasks from simple game example.................................................. 19
Figure 4.1 Summary of all assistance techniques discussed in this thesis.................................... 20
Figure 4.2 Fog of War ................................................................................................................... 23
Figure 4.3 Bag search in World of Warcraft ............................................................................... 24
Figure 4.4 World of Warcraft proc icons ..................................................................................... 24
Figure 4.5 World of Warcraft new ability alert ........................................................................... 25
Figure 4.6 High particle density (left) vs. low (right) in World of Warcraft ............................... 26
Figure 4.7 Alternative icon placement in World of Warcraft with custom timers ....................... 34
Figure 4.8 (Left) Cluttered UI in World of Warcraft ................................................................. 39
Figure 4.9 (Right) Minimalistic UI in World of Warcraft .......................................................... 39
Figure 4.10 Character nameplates in World of Warcraft ............................................................ 43
Figure 4.11 Quake 3 (L) Character model color change (R) Default model example .................. 44
Figure 4.12 the Agency of Anomalies [L35] ............................................................................... 58
Figure 4.13 PUBG [L36] screenshot of the same scene with different graphics settings .......... 60
Figure 4.14 Path guide to a defense point in Overwatch ............................................................ 62
Figure 4.15 Ability (Cut) usage reminder in Pokémon (Fire Red / Leaf Green) ......................... 91
Figure 4.16 Floor guide to a defense point in Overwatch ............................................................ 62
Figure 4.17 Difficult Decisions – from Life is Strange ............................................................... 89
Figure 4.18 Hint used in Golden Trails ...................................................................................... 92
Figure 4.19 City guard dialog showing points of interest ........................................................... 96
Figure 4.20 (Right) Witcher 3 World Map ................................................................................. 98
Figure 4.21 (Left) GTA V World Map ......................................................................................... 98
Figure 5.1 (Left) Top-down view. (Bottom) Third-person view. (Right) Sniper ......................... 105
Figure 5.2 (Left) Candy Crush Saga. (Bottom) Diablo 3 (Right) Sniper Elite ......................... 106
Figure 5.3 Highlights – Top-Down (Left) Highlights – Third-Person (Right) ......................... 110
Figure 5.4 Hint: Top-Down (Left) Hint: Third-Person (Right) .................................................... 110
Figure 5.5 Compass: Top-Down (Left) Compass: Third-Person (Right) ................................. 111
Figure 5.6 Mean score for each game and technique combination played ............................... 114
Figure 5.7 Mean time (seconds) per successful click (Target Found) for both games ............... 115
Figure 5.8 Player agreement ratings for Top-Down game by Assistance Technique ............... 117
Figure 5.9 Player agreement ratings for Third-Person game by Assistance Technique ........... 117
Figure 5.10 Player agreement ratings for assistance effectiveness and usefulness ................... 118
Figure 5.11 Assistance Technique Rating – Top-Down (Left) Third-Person (Right) ............... 119
Figure 5.12 Best fitting technique for each game according to players .................................... 120
Figure 6.1 Highlight Variations in both games ........................................................................... 130
Figure B-1 Questionnaire: Pre-Experiment .............................................................................. 153
Figure B-2 Questionnaire: Post Technique ............................................................................... 153
Figure B-3 Questionnaire: Post Game ...................................................................................... 153
Figure B-4 Questionnaire: End of Experiment ......................................................................... 153

viii
1 - Introduction

Video games can be challenging, and such challenges are part of what makes a game stimulating and entertaining for the player [76]. If the game is too difficult, however, it can soon become frustrating [25]. On the other hand, if the game is too easy, players may become disengaged and uninterested. The latter is a prominent issue in multiplayer games wherein a higher ‘skill gap’ between participating players can lead to frustrated novices and unengaged experts [29].

Games try to overcome such problems by either helping struggling players or hindering skilled players. Examples of this can be found in traditional (non-video) games or sports, such as giving players a head-start in racing or score-based games, or having skilled players use their less-dominant hand in tennis. Similar examples can be found in video games where less-skilled players are given assistance in a number of ways; for example, players could be given higher hit points (health or life) in competitive fighting or shooting games, special “catch-up” powers in racing games, or hints and tips in strategy or puzzle games.

There are many ideas such as the ones briefly described above, most of which are dependent on the context of the game being played. In this case, ‘context’ refers to situational characteristics around playing the game, such as the number of players, whether there is a score or not, what players can do, etc. For example, giving a player a head start with points requires the game being played to have a scoring system to begin with. Such a technique would not translate to a racing or puzzle game that does not keep a score. Such ideas, methods, or strategies for adjusting player performance in video
games are what we call *Assistance Techniques* and that is how we will be referring to them throughout the thesis.

Many of the techniques we discuss in this thesis have been explored in one way or another in other academic work, while others have been implemented in some commercially available games. With this work, we hope to bring together all the relevant information for assistance that developers and designers might need in one place.

This would have value to designers, as many – if not all – games these days are complex and varied enough to include more than one fundamental action (core task). That would mean a lot of work for someone hoping to find information pertaining to assistance for the tasks in their game. This is because most of the previous work in this area focuses on one core task such as pointing or steering, and a designer building a game with both would have to look through multiple sources for the information they seek.

In this thesis, we propose a more generalized way of discussing assistance in video games, regardless of the kind of game being developed. By looking at the fundamental actions players perform in games – *core tasks* – such as signal detection, reaction time, or pointing, we can talk about an idea, a method, or a *technique* that can provide such an effect as described earlier, while still being able to translate that idea to any game that also includes the same fundamental action.

We have identified 10 unique *core tasks* of gaming and discussed the process of identifying those tasks. We have included sufficient detail on the process to help
designers identify these tasks in their own games, to better choose the appropriate assistance techniques.

To demonstrate the effectiveness of our generalized model, we built three different games that share a common core task. For our demonstration, we chose Visual Search as the core task since it is one of the least academically explored from our list of core tasks. We successfully implemented three different assistance techniques in each of the three games, showing that even though the games are seemingly different (a puzzle-like game, a third-person adventure game, and a sniper simulation), the techniques can be adapted to fit all of them.

We then evaluated the effectiveness of these techniques by running an experiment on two of the three games: Top-Down (the puzzle-like game) and Third-Person (the 3D adventure-like game). The techniques tested were Highlights, Hint, and Compass. All participants also played the game with no assistance enabled at all. During the study we collected data on player performance, which included their final scores and the mean time of task completion. We also asked our participants to fill out several questionnaires to gauge their experience during the experiment and to get their opinion on the different assistance techniques.

The results of our experiment showed that all of the assistance techniques increased player performance to a certain degree. The best technique was Highlights, which more than doubled the average unassisted score for players of both games. Meanwhile, survey data showed us that all players preferred to have assistance enabled while playing, and would like to have the option to enable/disable it.
Our work is the first to compile the mass of knowledge of video game assistance in one place and to generalize how we discuss assistance for core tasks in games. We hope that our work gives designers a new language for discussing assistance techniques and an important starting point for making design decisions in games.

In short, the contributions of this thesis can be listed as:

- A definition of Core Tasks for video games with a comprehensive list of assistance techniques for each of the core tasks.
- A demonstration of how to analyze and identify core tasks in games.
- A demonstration of the versatility of the techniques by implementing and adapting the techniques into three different games.
- An experimental evaluation for three of the Visual Search techniques, showing how the various techniques can perform differently in separate contexts.

Organization

Given the nature of this thesis, we decided to distribute our references into three lists. The first is a traditional bibliographical list of “References”, including cited papers, journals, and articles. The second list contains all “Referenced Media”, including images, videos, and audio clips; such references are denoted by the prefix M [M#]. Some of the images referenced can be found within the thesis, whilst the remainder are simply referenced in-text. The final reference list is the “Ludography”, which lists all the video games that are referenced within this thesis, and are denoted by the prefix L [L#]. Multiple games are referenced throughout the document for comparison or demonstration.
The remainder of the thesis is organized into these chapters:

**Chapter 2 - Related Work**

This is a discussion of existing research and resources relevant to this thesis.

**Chapter 3 - Core Tasks and Assistance**

An introduction and discussion of both concepts.

**Chapter 4 - Assistance Techniques**

Each core task is introduced and explored at length. The assistance techniques for each of the core tasks are also introduced along with numerous examples to illustrate the concepts being discussed.

**Chapter 5 - Demonstration and Evaluation**

We present a practical demonstration of implementing the assistance techniques discussed in previous sections, specifically for 4.5 – Visual Search. This is then followed by an experimental evaluation and analysis of those same techniques.

**Chapters 6 and 7 - Discussions, and Conclusion and Future Work**

In chapter 6 - Discussion, we answer several key questions and clarify potential concerns relating to the application of Assistance Techniques in practice. Further, we explore and explain the reasons for the results of our evaluation. We then discuss possible paths that future research in this field may pursue to expand upon this research.
2 – Background and Related Work

2.1 - Difficulty

In their book *Rules of Play*, Eric Zimmerman and Katie Salen define a game as a “system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome” [23]. However this is by no means an exhaustive definition of what a game is - the book goes on to compare definitions from eight different sources before presenting its own, furthermore, there are many alternative definitions available on the Internet [24].

One commonly mentioned characteristic of games is the idea of conflict, competitions, and/or challenge. This is a critical aspect of games that can take many forms, from solo conflict against the game system to competition or cooperation between players themselves [23].

The fact that overly difficult games cause frustration, whilst easy ones lead to boredom, is considered “common knowledge” [25] amongst game designers. As such, how to design for the appropriate level of difficulty has become an increasingly popular subject with game designers [25], as well as what aspects of a game can be manipulated to control difficulty (Time Limits, Damage Scaling, HUD restrictions, etc.) [26].
2.2 - Flow

In psychology, cognitive flow is the state in which people are so immersed in an activity that everything else ceases to matter, and their perception and experience of time becomes distorted [27]. Flow is achieved when a person’s skill ideally aligns with the difficulty of the task at hand, which promotes a high level of engagement and focus on the task. If the task is too difficult, anxiety and frustration occurs – too easy, and boredom begins to emerge [28].

Ideally, game designers want their players to remain in a state of flow throughout their experience, which would represent a rich and meaningful engagement with the game [23]. To achieve this, Csikszentmihalyi outlines four task ‘characteristics’ that increase the probability of achieving a flow state. One such characteristic is to “demand actions to achieve goals that fit within the person’s capabilities” [28].

This refers to the balancing of player skill and game difficulty in such a way as to maintain equilibrium between stress/arousal and performance. Each player has a unique stress-performance curve, and thus a gradual increase in difficulty (easy, medium, and hard) is not always optimal. It has been suggested to use A.I. powered adaptive difficulty to dynamically change the game conditions to match a player’s skill level [28]. We might also take advantage of assistance techniques to influence the player’s performance and perceived challenge to correspond with an appropriate difficulty to achieve this cognitive flow [29].
2.3 - Assistance

One way of helping players who are struggling with a game challenge is to assist them with the task they are required to perform, increasing their effective skill. For example, suppose that a player is having a tough time hitting a target with 10 bullets. Instead of giving them 100 more to hit the target, we could instead assist them with their Aiming skill to increase the probability of a successful shot. There are several techniques that can be employed to achieve the desired result [29][30]. Previous work in this field demonstrates that providing assistance increases the overall performance of players and does not hinder them in terms of skill development [31].

Many research papers have explored the concept of assistance from different standpoints. Some examples of these include measuring performance due to camera viewpoints in racing games [35], balancing multiplayer games for racing [36], using assistance in shooting games [37], or comparing the effectiveness of several visual search assistance techniques in an artificial reality setting [38]. These papers provide valuable information comparing different techniques [39] or the parameters within the same one [38]. However, they are almost always looking at it within a particular context – specific input or display devices, a certain type of game, etc. – which makes it difficult for game designers to find the information they need for their projects without having to search through several sources. Our goal is to discuss assistance techniques at a general level that could be applied to any game, irrespective of context. We will refer to previous work in more detail wherever applicable as we discuss assistance techniques further on.
3 - Core Tasks and Assistance

3.1 - Core Tasks

The question we needed to answer is: *How can assistance techniques be discussed and described in a general way without needing to know the context of the game?* To do that, we need to find something common between all games. What actions do players perform while playing games, regardless of *what* they are playing? We can say that players *notice things*, they *remember* certain things like locations or story elements, they *react* to something that happens in the game by pressing a button or moving a joystick, and so on. So, if we can find a way to assist them to notice/remember/react better, then that technique could be applied to *any* game that requires the player to notice or remember something.

These terms, however, are quite vague. We need a better and more structured way of discussing these ‘fundamental actions’ in video games to be able to have a constructive discussion on how to provide them with assistance. For this purpose we describe *Core Tasks*. A list of fourteen core tasks was established to help with a gamification of calibration problem, where they had a similar goal of trying to find the fundamental tasks in games to be calibrated [1]. We took that list as a starting point, and after analyzing it we ended up with a modified version of ten core tasks. The details of how some of the original items in the list were merged are discussed as each core task is explored further.
1. **Signal Detection:** Perceiving a stimulus, like sound, or light.

2. **Signal Discrimination:** Distinguishing between audio/visual stimuli.

3. **Body Controls:** Using *Muscle Activation* such as flexing, *Movement* of a body part, or *Ambidexterity* – the use of two hands.

4. **Reaction Time:** Reacting to a perceptual stimulus as quickly as possible.

5. **Visual Search:** Finding a visual target in a field of distractors, also includes *Pattern Recognition* (determining the presence of a pattern amongst a field of distractors).

6. **Aiming:** Accurately pointing at a target (possibly using a device), without feedback about the pointing direction.

7. **Pointing:** Accurately pointing at a target with feedback about current pointing position.

8. **Steering:** Moving or guiding an object along a path.

9. **Memory:** Memorizing and/or retrieving sets of items, sequences, and/or mappings.

10. **Spatial Memory:** Remembering the location of items in a space without persistent visual cues.

To make sure that in-game tasks can be mapped to the list of core tasks above, we analyzed a number of the most popular games from a variety of genres. The list of games was generated using the high level taxonomy of genres on Wikipedia [22].

Examples of core tasks observed in games include: in *Super Mario Bros.* *Aiming* when Mario has the fire flower power up and shoots a fireball to hit an enemy; *Reaction Time* when jumping over turtle shells or dodging the throws from the Hammer Bros.; and, *Steering* for moving Mario along the path between obstacles and onto platforms. Modern games are much more complex and could potentially include all the core tasks like in Grand Theft Auto or the big first-person shooter games such as the Call of Duty franchise. In such games, shooting is *Aiming* and *Pointing* when shooting at enemies. The games also involve *Steering* and *Spatial Memory* for navigating the environment, *Visual Search, Signal Detection/Discrimination* for noticing enemies and distinguishing them from allies, etc.

Our analysis of these games did not reveal any game actions or tasks that do not fit within the core tasks listed earlier. There are more games from the same or different genres and sub-genres that could also be checked in the same way, and we expect that they would all fit within the 10 categories of core tasks listed above.

### 3.2 - Assistance Techniques

#### 3.2.1 - Definition

We define *Assistance Techniques* as the methods used to help players as they perform core tasks.

The goal of these techniques is to adjust player performance as they complete a core task. For example, this can be achieved by affecting attributes such as their speed (e.g. altering cursor speed with Target Gravity), accuracy, or efficiency. These techniques may also be
used to maintain game flow where possible. Balancing performance using these techniques avoids having to alter the goals, tasks, complexity of the game, or the behavior of game elements such as opponents, the environment, etc. Perhaps most importantly, the only prerequisite for using one these techniques is the presence of its corresponding core task.

3.2.2 - Why explore Assistance in terms of Core Tasks?

Several research papers, like the ones mentioned in the Related Work section earlier, have already examined the effects and effectiveness of the application of some of the techniques we discuss here such as Sticky Targeting (slowing down the pointer) and Target Gravity (pulling the pointer towards the target) for Pointing [29].

Such works have been carried out within a context (a specific genre of game and/or input method) and normally tackling only one core task. This makes it difficult to reliably translate their findings to other types of games, and even harder for games that employ several core tasks, as the information would be scattered.

We propose a more generalized way of discussing assistance techniques that ignores context (genre, input medium, theme, etc.), focusing instead on adjusting difficulty at the most fundamental level of play interaction (the Core Tasks). We believe that this could help us define core tasks more clearly, and perhaps help us discover new ways to provide assistance to those tasks. We also attempt to highlight gaps in the list of assistance techniques that either are not being employed by the gaming industry or have not yet been explored academically.
Game designers could use this framework to identify core tasks in their games and to choose which techniques would help them reach their desired level of difficulty.

### 3.2.3 - Difference between Assistance and Difficulty Adjustment

There are many ways of making a game easier for players. A game designer can choose which elements of the game can be changed to adjust difficulty, such as reducing the complexity of a challenge, or giving the player more power [26]. Such changes are normally context-specific like giving the player a straight path through a maze, or a more powerful weapon. Alternatively, Assistance Techniques as presented here aim to lower the difficulty by ensuring better performance in a given task without necessarily changing any aspects of the challenge presented by the game.

As an example of difficulty adjustment, suppose that a player is performing poorly in a moving-target shooting task. One approach is to have the player score higher per successful hit. This way their score would go up even though their performance is unchanged. Alternatively, perhaps we could have the targets be stationary, making it easier to hit them. This would again give the player a higher score, but it did not necessarily improve how well they shoot. In this case, the player’s ability to shoot is the main performance metric as stated in the definition of assistance techniques earlier.

Those ideas essentially alter the design of the game, such as reducing the challenge by modifying the scoring system or changing the behavior of the targets to stop moving, to give the player an advantage. However, we cannot always rely on such ideas because some of these changes to the game design may not be preferable. For example, if the
same task was found in an alien invasion themed game, it would be nonsensical to have the aliens (targets) stand still and take a hit.

Using assistance techniques, we can discuss instead how to make the game easier without needing to know specifics of the game design. In the example above, shooting targets is a Pointing task. We can choose to employ one of the several techniques that assist with pointing such as Target Gravity or Sticky Targeting. This way, some of the shots that the player would ordinarily miss would now hit, increasing the overall accuracy without necessarily changing the goal of the game or the design of the challenge.

It is important to note that any of these methods of difficulty adjustment and any of the assistance techniques can be applied simultaneously. The purpose of this section is to highlight the difference between difficulty adjustment and assistance techniques in terms of implementation and their effect on the game so that designers can make an informed decision.

3.2.4 - Difference between Assistance and “Doing it for them”

In certain situations, a player may still perform poorly after assistance techniques and/or difficulty adjustments have been made. Any more assistance may very well be taking control away from the player and having the game play itself. For such situations, designers may choose to do the task for the player, either partially or even fully.

For example, in racing games, one of the options that designers often employ is Rubber Banding [40], where the player is given a boost to catch up to the other racers even if they are underperforming in Core Tasks such as Steering, Spatial Memory, or Reaction
Time. In shooting games, like the hypothetical one discussed earlier, designers could add in friendly non-player characters (NPCs) to help shoot some of the targets instead of the player [M13][41] if Pointing or Aiming techniques are insufficient.

It is up to the designers to decide when to provide such help, and to what degree. One of the reasons to give the players a boost like that is to make the game feel better. In competitive games this could help generate the feeling of a “close-call” game [42], which is exciting for all players, when otherwise the good players would have had an easy and boring victory, leaving the other players frustrated and disappointed. In single player games, “doing it for them” can help with keeping the player moving along the story or experience the designers built for them. Not doing so could result in the player being stuck for too long at a particularly challenging section of the game and risk them abandoning the game all together.

The ideas of how to “do” the task for an under skilled player can be design specific and it is up to the designer to find a suitable way to do it. However, we will discuss general ideas for when players underperform in a specific core task.

3.2.5 - Strength of Assistance Techniques

We define the strength of an assistance technique to be the intensity of its implementation, which relates to the parameters of a given technique. For example, the Target Gravity technique for assisting with Pointing can be considered strong with a high value for gravity and weaker with a smaller value.
Another example is the Companion Signals for the Signal Detection core task, which attempts to increase the probability that the players will notice the information that the original signal is trying to communicate to them. This is achieved by accompanying the original signal with one or more additional signals. If that technique is implemented with one additional signal, it would be considered weaker than an implementation with several additional signals.

For example, an implementation of Companion Signals with two additional signals would be considered stronger than an implementation with just one extra signal. The idea is that it is less likely that three signals would all go unnoticed at the same time; for example, due to obstruction, noise, etc.

3.2.6 - Modes of Delivery

Designers decide on where, when, and how these techniques are applied in their game. One tactic is to implement one or more of the techniques into the game by default, providing assistance in tasks that are not vital to the game, while giving little to none for the tasks that are supposed to be challenging. For example, in games like World of Warcraft, or Monster Hunter [L14], the players often encounter dangerous opponents whom the players need to defeat by performing Aiming and Pointing tasks, and perhaps have to React to the enemy actions. All those tasks could be assisted to a certain degree, or not at all. However, once combat is over and the player is rewarded with some treasure, the action of finding and picking it up should not be a challenging one. In that case, techniques for assisting with Visual Search and Pointing could be employed to their maximum capacity to make sure players find no difficulty in performing this action.
The assistance techniques can also be used in part to create different levels of difficulty for a game such as “Easy, Normal, and Hard”. Games sometimes use different names for their tiers of difficulty that may be more in line with the theme (Like “Recruit, Regular, and Veteran” in the military themed Call of Duty games), but the idea is that each level of difficulty is objectively more difficult than the last.

Designers can decide how many techniques to implement per level of difficulty, and how strong the technique implemented should be. A technique is stronger when its parameters are increased. For example, the Complementary Signals technique for Signal Detection applying 3-4 additional signals is stronger than an implementation with only 1 extra signal. In a shooting game, for example, “easy” mode warns the player of low ammunition by (1) playing an alert sound, (2) showing a 0/X text on screen, (3) the empty click sound effect when attempting to shoot without ammunition. “Normal” mode could then have a weaker version of this technique by removing the alert sound, and “hard” mode would remove the visual feedback on screen leaving the player to find out that they are out of ammo the hard way.

Assistance could also be provided dynamically. Designers can choose when and why assistance is given. One option is to provide assistance when a certain condition is met, such as losing on a particular level X number of times, or when a player’s performance is low, decided by some pre-calculated performance metric or with A.I. assistance. Another way is to give players the option to enable assistance, which could be a simple switch in a menu, or through some in-game system such as perks (sometimes also known as boosts, power ups, etc.), items and/or equipment, abilities, etc.
3.2.7 - Organization and Structure for Assistance Techniques

For each core task, we will list and discuss possible ways of providing assistance. Each technique will be described in terms of how to implement it and why it might be implemented. We will additionally identify a technique’s advantages and disadvantages, and highlight representative examples of the technique in published games. Any results or findings previously made will be referenced wherever applicable.

Several techniques discussed here have been collected from the different research papers discussed in the Related Work section earlier. Others were a result of brainstorming sessions and/or observations from commercial games that provide assistance to their players. We will mention more specific examples as we discuss each technique in more detail further on.

3.2.8 - Methodology

We started by searching for previous academic work that explored assistance in relation to one of the core tasks. For example, results that match the query for “Assistance in shooting games” could have useful information for the Pointing and Aiming core tasks, and perhaps for Reaction Time or Signal Detection.

The list of queries we used includes, but is not limited to, variations of the following on Google and Google Scholar:

- “Difficulty levels in games”
- “Balancing competitive games”
- “Effects of [Core Task] in games”
- “Difficulty curves and Flow in games”
To find the remaining assistance techniques, we looked at each core task individually and found games that incorporate that task. At this point, we had already verified our core tasks using the methods described in section 3.1 - Core Tasks. This meant that we were reasonably confident that any game we choose here would include one or more of the core tasks as shown in Figure 3.1. The games used for this were chosen from personal experience and from some of the more popular games per genre (Table B-1).

![Figure 3.1 Extracting Core Tasks from simple game example](image)

We then examined those games to find out how they may be affecting player performance in those tasks and any options or in-game abilities that affect performance as well. This analysis helped us verify any techniques listed or identify a new technique.

Our goal was to find, discover, and highlight the different possible techniques for all the core tasks. However, we believe that there could very well be gaps in our list. With this work, we hope to lay down the ground work for a comprehensive list of techniques that can be expanded and improved upon in the future.
4 - Assistance Techniques

In the following sections of this chapter, we will introduce each of the core tasks individually, along with their corresponding assistance techniques. Figure 4.1 provides a visual summary of all the techniques and relates them to each of the core tasks.

You may notice that some of the techniques listed in Figure 4.1 belong to more than one core task. These techniques share a similar name and behavior, but they are discussed separately for each of the core tasks to which they belong, to highlight any task-specific considerations and provide examples relevant to the core task.
Core Tasks & Assistance List Template

Each of the core tasks identified earlier will be explored in more detail in its own section along with all the assistance techniques related to that core task. In this section you will find a template that will be followed for each of the 10 core tasks. The template shows the organizational outline for each section and a brief description for each subsection.

4.X – Core Task

4.X.1 - Introduction

This is a short introduction to what this core task is, including how it may look in games and what difficulties players may find while performing it.

4.X.2 - Examples

Here we highlight examples from real games that incorporate this core task in their gameplay.

4.X.2.1 Examples of Core Task Assistance

This is an optional subsection that would showcase examples of games where the developers purposely provided assistance to players while performing this core task.

4.X.2.2 Examples of Player Exploits

This is another optional subsection similar to X.2.1, but instead showcasing examples where players exploit a feature of a game to gain an advantage in this core task or using third party software.
4.X.3 - Assistance Techniques

This is the list of all the techniques we have identified for adjusting player performance in this core task.

4.X.3.N – Assistance Technique N

Definition

This is a concise definition of one assistance technique (named N).

Generic Example

Examples are shown of how this technique may look or behave in a generalized way.

Game Example #Y (Default)

This is an example of a game where this technique is available to all players by default.

Game Example #Z (Optional)

This is an example of a game where this technique is available to players through some in-game mechanism such as an options menu, player abilities, power-ups, items, etc.

These may be manually enabled by the player or enabled automatically based on certain rules set by the game designers such as repeated losses, low performance, dynamic difficulty adjustment, etc.

Considerations

This is an optional subsection where we would discuss possible concerns, issues, or side-effects of implementing this technique that designers should be aware of. This is also where strength would be defined for techniques where it is not implicit.
4.1 - Signal Detection

4.1.1 – Introduction

Signal Detection is described as the “conscious perception of a particular sensory stimulus” [1]. This includes stimuli such as light, sound, and tactile feedback such as controller vibration. This is an important task in video games, as often the detection of certain elements may have a significant impact on the overall player performance [3]. For instance, a player who does not pick up on a ‘low ammunition’ signal from an on-screen icon could run out of ammo during a critical battle and lose the game.

4.1.2 – Examples

Visual signals use light (Figure 4.2), colors (Figure 4.3), symbols (Figure 4.4), and text or dialogue (Figure 4.5), to convey information to the player.

Figure 4.2 Age of Empires II’s [L51] Fog of War [78] is an example of using light in games
(Source: https://www.giantbomb.com/fog-of-war/3015-14/fog-of-war/3015-14/)
Figure 4.3 Bag search in World of Warcraft (Source: https://worldofwarcraft.com/en-gb/news/2817227)

Figure 4.4 World of Warcraft proc icons (Source: https://www.mmo-champion.com/threads/752372-Power-Auras-Effect-Welcome-the-Machine-Blue-Posts)
Music [M1], sound effects [M2], and dialogue [M3] are types of audio signals that can be similarly used.

**Tactile** signals are a type of haptic feedback, such as the controller vibration when driving on rough terrain (e.g. sand or grass) in racing games like Forza [L15].

### 4.1.2.1 Examples of Signal Detection Assistance

Certain games in the Call of Duty franchise feature optional _perks_ that players may choose to turn on during play. Some of these perks include Awareness, which amplifies the intensity and range of enemy footsteps, while Ninja mutes the player’s own footsteps to give a disadvantage to the enemy. What is interesting here is that to use these perks, players would have to give up other perks that they deem inferior, such as increased damage or health, or faster running and reloading.
4.1.2.2 Examples of Player Exploits

Some games may provide options that amplify or dampen certain signals, without the explicit intention of providing assistance to the player, but as a side effect of some other functionality. For example, World of Warcraft has settings sliders for Particle Effect Intensity, as seen in Figure 4.6. This option exists to allow the game to run smoothly on different strength computers. However, players often turn down particle effects to make certain elements in the game more noticeable, such as fire effects that are harmful, which can then be avoided more accurately.

Figure 4.6 High particle density (left) vs. low (right) in World of Warcraft

4.1.3 – Assistance Techniques

In this section we discuss the different methods that can be employed to make existing signals more or less effective depending on the end goal, whether it is to create an ‘easy’ mode, or to have a ‘hard’ mode that does not completely get rid of signals, or to dynamically provide assistance to players who are underperforming regardless of the current mode or difficulty. We have identified the following six techniques that can assist signal detection:
1- **Companion Signals**: Accompanying the original signal with an additional signal, often of a different medium.

2- **Augmented Parameters**: Modifying parameters of a signal to make it more or less noticeable.

3- **Announcements / Emphasis**: Feedback to point out a signal and/or for directing the player’s attention to it.

4- **Signal Priority**: Giving the signal more access to its media, such as more space on a visual display.

5- **Transformation / Replacement**: Altering the signal enough for it to be considered different from the original.

6- **Noise Reduction**: Modifying other elements in the game scene to make the signal more discernible.

### 4.1.3.1 – Companion Signals

**Definition**

Accompanying the original signal with an additional signal, often of a different medium, such as a visual cue for audio signals and vice versa. Companion signals aim to increase the ability of and likelihood that a player to notice the information carried by the signal.

**Generic Example**

An example would be a beep whenever a target appears on screen, or playing intense music when enemies are around which, could lead the player to pay more attention to visual signals to find them.
Game Example #1 (Default)

The 3 stages of the charged attack in Monster Hunter: World [L16] are conveyed to the player using a visual signal (pulsing energy around the character), an audio signal (a sound effect of varying pitch per tick), and a haptic signal (controller vibration with every tick). These three signals work together to make sure the player can time their attack effectively even if one of the signals is compromised. For example, if the monster they are hunting is thrashing and screeching, it can be difficult to rely on the visual or audio cues, but the vibrations can still be felt to time it correctly.

Game Example #2 (Optional)

The survivors in Dead by Daylight [L17] attempt to stay hidden and stealthy, but if they disturb some of the wildlife in the game, like a crow for instance, the crow would loudly fly away. A good killer would listen to the sound these crows make to find where the survivors are skulking. Some killers, however, find this a difficult task, and they may choose to enable the Spies from the Shadows [M5] perk that displays a visual signal alerting them to the location of the noise.

Considerations

An issue that may arise from implementing this technique is the over saturation of signals. If every signal is accompanied by one or more signals, one could effectively double or even triple the number of signals a player would need to be aware of, pay attention to, and understand.

It will be up to the designer to make a judgement call as to whether it is worth it to add more signals to their in-game elements. Alternatively, perhaps a different technique could
be used if there are too many signals already, or by mixing and matching with other techniques; for example using the Noise Reduction technique for when there are too many signals running simultaneously.

The strength of Companion Signals is proportional to the number of additional signals implemented. It is important to remember that ‘strength’ here refers to the intensity of the implementation of this technique, not necessarily the effectiveness of the implementation. This is because it can be argued that after a certain number of companion signals, adding more would create too much noise and counter any positive effects of the original signals.

4.1.3.2 – Augmented Parameters

Definition
Modifying the intensity of existing signals. For instance, one or more parameters such as brightness, contrast, or saturation for visual signals, or volume, pitch, or range for audio signals could be modified to reach the desired effect. The intent here is to make the original signal more salient so that the player is able to notice it more reliably.

Generic Example
In a situation where a player is running low on a certain resource, such as health, energy, or ammunition, a sound could be played to alert the player. This sound could be made louder to ensure that the player notices the depleting resource in time.

Game Example #1 (Default)
One of the characters in Overwatch is Lucio, who uses a Sonic Amplifier [4] that fires a burst of soundwaves as projectiles. Each shot of Lucio’s weapon is accompanied by a
sound effect. The pitch of this sound effect increases with every shot made [M8]. Experienced players are able to detect the change in pitch and predict when they may need to reload in advance, without needing to take their eyes off the center of the screen - where most of the action happens - to check the ammunition count in the corner of the screen.

**Game Example #2 (Optional)**

In Dead by Daylight, when a survivor is hurt, they bleed onto the ground. Killers who notice the blood on the ground, could follow the trail to find where the survivor may be hiding. This may be a difficult task as a spot of blood on dirt in a dark environment may not be the easiest thing to notice. One of the perks the killer may choose is Bloodhound [M6][M7], which turns the color of the blood on the floor into a bright crimson red that is much more visually salient.

**Considerations**

It is important to be aware of what could happen when a parameter of a certain signal is modified. For example, if we increase the volume of an audio signal, it could potentially drown out other sounds.

The strength of this technique depends on the type of signal, and the property of that signal that is being modified. For example, the louder an audio signal is, the stronger it is. For pitch, however, strength refers to the biggest change to the signal’s phenotype. For instance, a drop of 100Hz has a more noticeable change than an increase of 3Hz, and thus would be considered as the stronger implementation even though the pitch was reduced.
4.1.3.3 – Announcements, Explanation, and Emphasis

Definition

Feedback to point out a signal and/or for directing the player’s attention towards the signal. The idea here is to make the signal very obvious to the player, or simply explain it to them.

Generic Example

A stronger implementation of this technique may be a one-time tutorial to explain certain signals, or perhaps if a player has repeatedly missed a signal.

For example, a large message might appear across the screen stating “Enemies nearby” when battle music plays, or if the enemies appeared on screen but the player has not noticed yet. Alternatively, perhaps a friendly non-player character (NPC) would explain a sound effect by saying something along the lines of “When you hear the sound, it means you are low on energy, so pay close attention.”

Game Example #1 (Default)

To successfully complete a Mounted Attack in Monster Hunter: World, the player must perform certain actions based on the monster’s movements. Aside from the monsters’ behavioral cues that indicate what the player should do, visual signals are displayed in the top right corner of the screen that tell the player what to do and which button to press. This is where this particular technique comes into play; the first time a player mounts a monster, a message telling the player what to do is displayed at the center of the screen while the signal in the corner of the screen glows for more emphasis [M9].
**Game Example #2 (Optional)**

In *World of Warcraft*, one factor that determines how well a player performs during a raid is their ability to notice enemy attacks and react quickly appropriately. Most bosses have unique abilities, and are designed to give players a hint, one way or another, so that they have enough time to react. For instance, a certain enemy may shout out a moment before unleashing a devastating attack, or perhaps certain sound and/or visual effects are used as a build up to a powerful ability. Players are expected to pick up on those little hints and cues and learn the bosses' combat patterns to defeat them.

One of the most popular add-ons is *Deadly Boss Mods* [8], which displays essential information such as when the next ability is going to happen, what the player should do, etc. for every possible boss encounter in the game. This helps players notice abilities and react to them properly much more effectively. In fact, this add-on is so effective that players are unwilling to group-up with others who do not have it installed [9].

**Considerations**

This technique can be very effective, therefore, it is up to the game designer to decide whether this technique is necessary, and if so, carefully choose when, where, and how it is to be implemented.

The strength of this technique depends on the intensity of its implementation, duration, and frequency. For instance, a one-time only message on screen notifying the player of a signal when it appears for the first time would be considered a weaker implementation than an extended dialogue recording that is played often to explain a signal to the player.
4.1.3.4 – Signal Priority

**Definition**

Media that transmit signals have limitations in the real world. Visual signals are limited by the dimensions and resolution of their display. Auditory signals may overlap and create too much noise, or perhaps drown out other sounds if they are too loud. Even haptic signals can be limited depending on how many parts can move simultaneously.

A signal can be given higher priority than others to ensure it is detected. Giving a signal higher priority is achieved by giving it more access to its media, such as more space on a visual display, louder volume, or longer uninterrupted vibration patterns.

**Generic Example**

An icon indicating diminishing resources may blink in the corner of the screen, while a large message in the center of the screen is less likely to be missed. For example, the Blue Screen of Death [10] is nearly impossible to miss since it takes up the entire screen compared to a temporary notification. A similar example from games would be the Game Over screen [77] which often covers most of the display.

**Game Example #1 (Default)**

Whenever a game has a character than can be killed, or an object than can be destroyed, it is not uncommon to find a variation of a *health bar* [12], like the ones often found in arcade fighting games such as Street Fighter or Tekken. Call of Duty is a first-person shooter where the player character can die and the player’s goal is to stay alive as long as possible, but they do not have a health bar. Instead, when the player is injured, the screen would get bloodier or have a red hue overlaid around the edges [M10]. This red overlay
conveys to the player that they are hurt and should retreat, hide, and recuperate. This is a good example of using up plenty of the visual display real-estate to guarantee the player notices. The color overlay is often transparent as well, which counteracts some of the issues that may arise from using up this much space for one signal.

**Game Example #2 (Optional)**

Most characters in World of Warcraft have abilities with *cool downs*. This means that the player must wait a specific amount of time before being able to use that same ability again. By default, once an ability is used, its icon would be overlaid with a dark tint that is slowly wiped away clockwise for the duration of its cool down. Players may have multiple abilities, each with its own icon. These icons are small and placed near the edges of the screen by default. This can make it difficult to know when an ability is usable again, especially during hectic fast-paced gameplay.

There are many add-ons that move the ability icons to somewhere more noticeable (Figure 4.7) such as the center of the screen. Some of these add-ons even enlarge the icons, so that they are easier to see.

![Figure 4.7 Alternative icon placement in World of Warcraft with custom timers.](https://i.imgur.com/QXMlK.jpg)

(Source: https://i.imgur.com/QXMlK.jpg)
**Considerations**

Unlike the Augmented Parameters technique, Signal Priority is not simply about modifying a property of the signal, but it is rather about understanding the limited media real-estate and distributing it effectively. It is also not simply about the percentage of real-estate given to each signal. For example, the same size of visual space on a display would be worth more at the center of the display compared to the same size at the corners. The value of the media real-estate depends on the type of medium and the goal of the signal and it is up to the game designers to analyze what they are working with and make informed decisions with that in mind.

It is also important to remember that signals are not the only contenders for media real-estate. In most games it would not make sense to use up 100% of the display for signals as that would leave no space for the game itself, like in Figure 4.8 shown for Noise Reduction.

The strength of this technique depends on the amount of media real-estate [11] used up by the signal. For example, a visual signal using 100% of the visual display would be considered a much stronger implementation than a signal using up a small portion of the given display. Similarly, a sound effect at maximum volume or using up all audio outputs would be considered a strong implementation.
4.1.3.5 – Signal Transformation / Replacement

Definition

Unlike the Augmented Parameters technique, the goal of this technique is to alter the signal enough for it to be considered different from the original. This could be as simple as changing the color of a visual signal, or using a completely different sound effect.

Generic Example

An example would be using flames instead of an icon to portray an activated power up for a character.

Game Example #1 (Default)

In the Call of Duty franchise, a first-person shooter, players have access to kill streaks [13], which are rewards given to players during a match for getting a certain number of consecutive kills without dying.

One of the most common kill streaks that has been around since this concept has been introduced is the Unmanned Aerial Vehicle (UAV) [14]. The UAV displays the location of enemy players on the mini-map for a short period of time. Enemies are represented by a red dot on the map.

Much more recently, a new version of this killstreak was introduced called Advanced UAV [15] (formerly known as Directional UAV). This Advanced UAV serves a similar function, but instead of dots, it uses arrows to represent enemy players’ location and direction of movement on the mini-map. Not only is this a different signal than before, it also carries additional information.
**Game Example #2 (Optional)**

In Overwatch, like most first-person shooters, players get a reticle or crosshairs at the center of the screen to help them aim more accurately. Overwatch also gives the player the option to customize their reticle, including its shape and color. Players then choose a custom reticle that is best for them, which superficially seems like a simple cosmetic customization feature. However, players seeking to improve their performance at the game are often advised to modify their reticle, specifically to a bright color like lime green [M11]. The idea is that the green color is more likely to stand out amongst all the colorful character and abilities happening seemingly chaotically on screen and makes it easier to quickly pinpoint the reticle and check if the player is aiming accurately or not.

**Considerations**

When changing a signal, it is important to be aware of the information the signal is trying to deliver to the player and to make sure nothing is lost in translation. The game designer wishing to implement this technique should be able to answer questions such as *what does the original signal tell the player? Does the new one convey the same information? If not, is something missing, is it being misinterpreted, or perhaps is it giving away more information than before?*

This technique is a little more difficult to discuss in terms of strength as it is not a quantifiable change, and some changes are not necessarily “better” than others. In this case, a stronger implementation of this technique would be the one that has the most drastic change to the original signal. For example, changing a red dot to a green hexagon is weaker than changing the red dot to a miniature animation of planet earth.
4.1.3.6 – Noise Reduction

Definition

Modifying other elements in the game scene to make the signal more discernible, such as muting certain sounds, or dimming the lighting.

Generic Example

Examples would be desaturating everything on screen except for the signal, or muffling game sounds to make sure character dialogue is heard clearly.

Game Example #1 (Default)

Whenever someone speaks using the in-game voice chat in Counter Strike Global Offensive [L18], the volume of everything else is reduced slightly so that the voice is loud enough to be heard. A similar feature is found in some common communication software used by gamers such as Discord [L40] and Skype [L41].

Game Example #2 (Optional)

World of Warcraft offers multiple User Interface (UI) customization options to players. The game UI can easily become cluttered line in Figure 4.8 with all the ability icons, chat boxes, status bars, text notification, etc. Players can choose to fiddle with these options to create a sleek, minimalistic UI like in Figure 4.9 that fulfils their in-game needs while reducing the amount of noise on screen.
Considerations

Designers need to take care not to remove something that may be considered noise in some contexts but might be vital to the game in some way. For example, UI elements may be momentarily removed so that elements of the game environment are easier to see. It is conceivable that some of those UI elements are important, such as player hit points.
4.2 – Signal Discrimination

4.2.1 – Introduction

Signal Discrimination is described as determining that there is a difference between two stimuli (e.g., two colors or two sounds) [1]. Being skilled in signal discrimination in games can give players an advantage [39]. For example, noticing an enemy hiding between allies, or an ability being performed by an enemy in a crowded battle, would provide a player with an edge over competitors who have more difficulty with such tasks.

4.2.2 – Examples

Visual, Audio, and Tactile signals such as those described in Signal Detection are the focus here. However, signal discrimination assumes that signals do not need to be or have already been detected, and it is the differences within one category of signal that is the main task requirement.

4.2.2.1 Examples of Signal Discrimination Assistance

An example of Signal Discrimination in games is the Color Blind options that some games provide. Normally, a toggle-able feature is available that swaps out the default game colors for those that are easier for color-blind people to distinguish.

An example of this in practice can be found in League of Legends [L19], a multiplayer team-based competitive game where each team has a unique color for its characters, building, and map icons. Being able to distinguish friend from foe on the map is very important for players to make decisions as to where to go or what areas to avoid during the course of the match.
4.2.3 – Assistance Techniques

Being able to understand and distinguish incoming signals is essential for making correct decisions or making those decisions more quickly in time sensitive games.

1- **Additional Cues:** Presenting additional signals to convey complementary information.

2- **Augmented Parameters:** Modifying parameters of a signal to make it more or less noticeable.

3- **High Contrast:** Transforming/replacing a target signal to vary the contrast between it and other signals.

4- **Noise Reduction:** Reducing noise to better notice certain details about the target signal.

4.2.3.1 – Additional Cues

**Definition**

Presenting one or more additional signals to convey complementary information about the original signal detected. This additional information can help the player distinguish the original signal from others that may be similar.

**Generic Example**

Adding extra signals to complement the existing ones can help highlight the differences. For example, in some shooting games, when pointing at another player, the cursor/crosshairs may change color depending on whether the character being pointed at is friend or foe. Alternatively, perhaps an outline would appear around the character. This
is especially useful when having to distinguish between characters or signals that may be similar such as soldiers that may have similar outfits but are on different teams.

**Game Example #1 (Default)**

First-person shooters, such as Call of Duty or Overwatch, display a red outline around enemy players when they are targeted. This helps players quickly distinguish between targets as it is possible for players on both teams to look nearly identical.

**Game Example #2 (Optional)**

One of the UI options that players can enable in World of Warcraft is the *character nameplates*. Enabling character nameplates would display the name of every character above it along with its health bar. The nameplates are colored differently depending on whether the character is hostile or not. Players also have control over certain filters such as showing nameplates for hostile characters only. This would make it easier to quickly notice enemies on screen. Figure 4.10 shows the nameplates in action, in which the green ones indicate non-player characters (NPCs) to make it easier for players to find interactive in-game characters such as vendors or quest givers.

**Considerations**

Similar to the Companion Signals technique for Signal Detection, it is possible to accidentally have too many signals and clutter the display. This is a problem that can often occur in World of Warcraft if the options are enabled as described in Game Example #2 above leading, to something similar to Figure 4.10 where the nameplates overlap making it difficult to tell them apart.
4.2.3.2 – Augmented Parameters

**Definition**

Similar to the assistance technique described for Signal Detection, amplifying or weakening certain parameters of a signal can make it more or less noticeable compared to other similar signals.

**Generic Example**

If several similar-looking targets appear on screen, the next correct target could be made brighter or larger to distinguish it from the others.
**Game Example #1 (Default)**

Overwatch has unique footstep sound effects for each playable character. This allows players to differentiate characters without having to see them. Furthermore, players on both teams may choose the same character, so being able to detect the sound may not be enough to tell the player whether it is a friendly or enemy player. To make the task of **discriminating** between the two easier for players, the developers decided to make enemy footsteps much louder than those of teammates.

**Game Example #2 (Optional)**

Expert players in Quake 3 [L42] found ways to change the colors of character models in-game [54]. Green was their preferred color as the human eye is best capable of differentiating the varying shades of green.

Figure 4.11 below shows the drastic difference this makes in game.

[Image]

Figure 4.11 Quake 3 (Left) Character model color change. (Right) Default model example.

https://gpstatic.com/acache/29/68/1/uk/s3-6c077539a04289affe0d09e4b0e99d8b.jpg
Considerations

This technique may rely on having multiple similar signals. For example, increasing the volume of a particular signal might not give the player any useful information if there is not a quieter one to compare it to.

4.2.3.3 – High Contrast

Definition

Transforming or replacing the signal to vary the contrast between it and other signals, thus changing the difficulty for differentiating them from each other.

Generic Example

Signals with higher contrast are most likely to be easily distinguishable. For example, it would be harder to distinguish between two cars in the distance where one belongs to a team member and the other to an opponent, compared to trying to distinguish between human speech and a car’s engine.

Game Example #1 (Default)

Each character in Overwatch can perform an ultimate ability, which is accompanied by a unique battle cry. The battle cry heard would be different depending on whether the character is an ally or an enemy [M19]. Experienced players can distinguish between the two versions and know whether they are in danger and should hide, or it is a friendly player using their ultimate ability and perhaps it is time to attack as well.
Game Example #2 (Optional)

During a match of Rocket League [L34], player nameplates appear above the cars they control. This helps players locate other players are around them and distinguish between teammates and opponents since the nameplate color reflects the team to which they belong. This can be difficult to do sometimes especially in environments with similar colors to the nameplates [68]. Rocket League provides players with a “High Contrast Nameplates” option [M22] that enables an implementation of this technique.

Considerations

Unlike Augmented Parameters, this technique can convey information to the player with just one signal, especially if the transformation created high contrast between itself and other signals. The Overwatch unique footsteps would be a good example of this where players can identify the character without having to see them.

4.2.3.4 – Noise Reduction

Definition

Reducing the impact of unrelated signals can make it easier to notice certain details about the target signal that otherwise may have been overshadowed by noise.

Generic Example

Just like noise reduction was described for Signal Detection, we could similarly desaturate unrelated visual signals, or muffle auditory signals to help highlight the target signal(s) better.
Game Example #1 (Default)

When aiming through the scope in Sniper Elite [L26] the sounds of the surrounding battle are muffled and nearly muted to help players concentrate and line up a perfect shot [55].

Game Example #2 (Optional)

Part of OLDTV’s [L21] aesthetic includes visual effects that mimic the style of old CRT monitors. “Glitch” effects, which are similar to screen tearing, also appear when players are doing particularly well. Both of these effects can be optionally disabled by players. Some do this to avoid eye strain while others choose it to make it easier to distinguish the color/word combinations during the game and progress farther.
4.3 – Body Controls

4.3.1 – Introduction

This core task includes any interaction a player may have with the game that requires the use of the player body, in part or fully, as input. This may be in the form of motion or Body Movement, which refers to gameplay actions that are performed by moving one’s body or limbs.

This has been made popular with games on console systems, using special input devices such as Xbox’s Kinect or PlayStation’s Eye. This also includes movement in certain Virtual Reality set-ups that use hand-held controllers that match real world movement to the 3D world in which the player would be immersed.

Another way that body controls could be implemented is using Muscle Activation, which requires a person to activate one or more muscles to act as input (e.g. flexing biceps) [1].

There are games that use this non-traditional method of input, either to provide variety in entertainment or to give the ability to play to players that may not be able to use traditional input devices such as a game controller or keyboard and mouse. The distinction is that traditional input devices are essentially switches that can be activated in any number of ways, while the inputs of note for Body Controls are those that directly translate body movements to input using a variety of sensors.
4.3.2 – Examples

In a recently published paper, a game called The Falling of Momo was developed for training myoelectric prosthesis in amputees [66]. The game was played using a myoelectric ‘myo’ armband [67] that detects muscle contractions as input. The flexing direction and strength affected how the in-game character moved.

Commercial games such as Dance Central [L32] or Kinect Star Wars [L33] for the Microsoft Xbox Kinect used motion sensing for player movement and posing for specific game inputs.

4.3.3 – Assistance Techniques

Sensory input such as that of the Xbox Kinect can be more difficult to use compared to traditional input methods like game controllers, due in part to a lack of fidelity and accuracy. Sometimes this input may require a full body pose such as those used in certain dancing games.

1- Input Modulation: Amplifying or diminishing the input from sensors to reach a desired result.

2- Detection Threshold: varying the margin of error while posing or activating muscles.

4.3.3.1 – Input Modulation (Muscle Activation)

Definition

Manipulating input from sensors to reach the desired result and avoiding overexertion.
**Generic Example**

Certain actions can be amplified, such as moving the character faster with a smaller level of muscle activation, or perhaps reducing the speed at which in-game elements react to muscle exertion for games that require higher precision.

**Game Example #1 (Optional)**

In Momo, a prosthesis training game briefly described above in section 4.3.2, a calibration feature allows players with different muscle strengths and control abilities to effectively play the game.

**Considerations**

Unlike traditional input devices that always work in the same way once built, using the player as input would vary depending on the player’s physique. Therefore, it would be challenging to implement this technique in a way that is optimal for all users.

**4.3.3.2 – Detection Threshold (Body Movement)**

**Definition**

Certain games may require the player to physically replicate a full-body pose or an arm/hand position. The margin for error while posing may be varied to change the difficulty of completing this task.

**Generic Example**

If a player is expected to raise their arms up and outwards in a “T” pose, the detection threshold can be increased to accept poses that are closer to a “Y” instead of a “T”.
**Game Example #1 (Default)**

We were not able to find any examples of games where assistance was explicitly mentioned, neither enabled by default nor optional. However, it can sometimes be obvious that certain games that detect body movement have a very high detection threshold. In such games, players can progress even with minimal effort in terms of movement like in Kinect Star Wars [M21].

**Considerations**

By implementing this technique, it may be possible to reduce the total number of detectable poses.

For instance, in the generic example above, if Y-shaped poses are to be accepted for T-shaped pose input, then T and Y poses are no longer two distinct poses that could have otherwise been used as input for different purposes.
4.4 – Reaction Time

4.4.1 – Introduction

Reaction Time refers to the duration of time measured from the moment the player is presented with a certain stimulus (which can be visual, auditory, haptic, or information) to when the player responds [1]. A reaction time task requires players to react as quickly as possible, possibly following Signal Detection or other task.

During that time, the player processes what has happened, identifies the meaning of the stimulus, and makes a decision as to what to do next (which includes doing ‘nothing’).

4.4.2 – Examples

There are many games that present the player with situations that require decisions to be made under tight time-constraints. Blocking an incoming enemy attack in a fighting game, turning away from an obstacle in a racing game, and clicking at the appropriate time in rhythm games, are examples of this.

This is even more vital in multiplayer games, specifically competitive ones such as the popular Multiplayer Online Battle Arenas (MOBAs) [73] like League of Legends. Players even go out of their way to train their reaction time [M14] to become better at those games and increase their rank.
4.4.2.1 Examples of Reaction Time Assistance

Guitar Hero [L20] is a game based almost entirely on speed and reaction time. Using a special guitar shaped controller with buttons in place of strings, players have to press those buttons as they appear on screen and strum the guitar simultaneously.

Guitar Hero provides the player with several difficulty modes ranging from Beginner and Easy to Hard and Expert. In the easier modes, the shapes on screen move significantly more slowly, and there are fewer buttons that the player is expected to press. In the Beginner mode they go even further by allowing the player to pass simply by strumming regardless of which button was pressed, and the penalty for making mistakes is drastically reduced.

4.4.3 – Assistance Techniques

Most of the techniques discussed for Signal Detection can also be used to help with Reaction Time. Our goal with Signal Detection was to get the player to simply notice the signal without regard to how long it could take. In this case, however, the time it takes for the player to notice matters, and as such, when implementing any of the techniques it is important to test and measure its effects on Reaction Time.

1- **Additional Cues:** Additional signals to alert the player that they are required to react.

2- **Advance Warning:** Notifying players that a time-sensitive action is imminent.

3- **Reaction Window:** Adjusting the duration window for which a successful reaction can be made.
4.3.1 – Additional Cues

Definition
Adding additional signals to alert the player that they are required to react. Some players may be quicker at detecting certain types of stimuli than others, or perhaps another cue would be helpful if the original alert was drowned out by noise.

Generic Example
An example of this would be responding to an enemy attack once the player hears their attack charging up before seeing it happen.

Game Example #1 (Default)
In rhythm games such as Guitar Hero or Dance Dance Revolution [L22], players can coordinate their button presses with the beat of the music instead of relying on tracking the on-screen indicators.

Game Example #2 (Optional)
Deadly Boss Mods (DBM) [8] is one of World of Warcraft’s most popular third-party add-ons. Once enabled, DBM will trigger additional effects, both visual and auditory, to alert players of certain game mechanics such as powerful boss abilities. This add-on provides such an advantage that many groups that join to play together (aka Guilds) would reject members that do not have it installed [69], as the player would not be able to perform at a satisfactory level.

Considerations
A potential issue that may arise is the over-saturation of signals. Game designers must be
cautious when adding additional signals to their game, especially if that is to be repeated for multiple game features that may be active simultaneously.

4.4.3.2 – Advance Warning

Definition

Announcing to the player in some way (text, icons, sound effects, etc.) that they will be required to perform a time-sensitive action momentarily. This can give players the time they may need to prime themselves for an upcoming reaction.

Generic Example

An example would be a countdown before a certain button needs to be pressed. This can give the player time to identify the button and prepare to press it quickly as soon as the countdown is over.

Game Example #1 (Default)

While performing certain actions in Dead by Daylight such as repairing or healing, players can be presented with a Skill Check [16], which is a Quick Time Event (QTE) [17] that requires a quick and accurate response. If they miss the skill check, they would lose some of the progress they have made and alert the killer to their location. Therefore, successfully passing skill checks is essential for success.

The skill check is announced via a gong-like sound effect shortly before it appears on screen. This warns players to pay attention to where the skill check will appear and prepare to respond. Killers also have access to optional perks that affect this, such as the Hex: Huntress Lullaby [18] which reduces the time between the warning and the skill
check. In its strongest form, this perk removes the warning entirely, making it very
difficult for players to successfully hit their skill checks.

**Game Example #2 (Optional)**

One of the third party add-ons for World of Warcraft is called Deadly Boss Mod (DBM).
During difficult boss encounters, DBM generates an announcement (auditory and/or
visual) to warn players of an impending attack or ability. This is sometimes paired with a
count-down, which gives players enough time to re-position their character or prepare
their next move.

**4.4.3.3 – Reaction Window**

**Definition**

Certain game mechanics require the player to respond within a limited timeframe.
Responding before or after can produce a negative effect or nothing at all. By adjusting
the duration of this time ‘window’, we can vary the difficulty of this task while
maintaining the pressure of a quick time event on the player.

**Generic Example**

A player may need to manually block an incoming attack within a second. Increasing the
time for the player to react would make that task easier and vice versa.

**Game Example #1 (Optional)**

The Nurse [70] is one of the more difficult killers to play in Dead by Daylight. Her
unique main ability, called Blink [71], allows players to teleport forward very quickly.
Once a blink is completed, players have a small window of opportunity to launch a
second blink to either gain more distance or flank a player. Once that moment has passed, however, the Nurse is stunned momentarily and unable to move.

The timeframe of this window is quite narrow, and it is common for players to get stunned while still attempting to line up (aim) their second Blink. The game provides players with in-game add-ons for the Nurse’s power that increase the time before the Nurse is stunned [72].

**Game Example #2 (Optional)**

The skill checks from Dead by Daylight require the player to press a button when a pointer moving clockwise along the circumference of a circle reaches a highlighted portion of the circle. There are several perks [19][20][21] in the game for both killers and survivors that affect the size of the highlighted area. An increase in the size is considered an advantage to survivors as it is easier to hit those areas and they have more time to do so accurately.
4.5 – Visual Search

4.5.1 – Introduction

Visual search is the task of finding a visual target within a “field of distractions” and/or noise, similar to the goal of *Where’s Waldo (aka Where’s Wally)* [74], where the player is instructed to find a uniquely dressed character in a large image of a highly colorful scene populated with many other characters and objects.

When assistance techniques for Signal Detection were discussed earlier, the goal there was to assist the player in *noticing* the signal (likely while other tasks are being undertaken). In Visual Search, however, it is the player’s goal to actively search for and find a visual signal.

4.5.2 – Examples

There are games that implement Visual Search as the main focus of gameplay such as the ‘Hidden Object’ games that have a similar premise to *Where’s Waldo*. Players are shown an image of a scene and given a list of objects to find, like the game in Figure 4.12.

Figure 4.12 A ‘hidden object’ game, called the Agency of Anomalies [L35] demonstrating the Visual Search core task. (Source: https://www.unigamesity.com/best-hidden-object-games-of-2012/)
Another way Visual Search can take the spotlight is with ‘Find the Difference’ games where the player is shown two or more copies of an image and they would have to find a certain number of subtle differences between them. Jigsaw puzzle games may also include a Visual Search element to them depending on how the pieces are presented to the player (e.g. scattered all over each other).

Aside from games intentionally including Visual Search challenges, many games in one way or another require the player to look for something. Examples are, finding an enemy in a crowd, finding an item in an inventory screen, or even finding the correct piece to move in Candy Crush [L24].

**4.5.2.1 Examples of Visual Search Assistance**

Many of the hidden object games, such as Grace’s Quest: To Catch an Art Thief [L43], provide the player with a hint button to use in case they get stuck at a particularly difficult part of the game. The number of times this can be used is usually controlled to avoid making the game too easy. The details of the hint vary from game to game, but they would all follow one or more of the assistance techniques discussed here.

Also, in World of Warcraft, players have an inventory of items, each represented by an icon. The inventory is expandable and can accommodate over 100 items. When full, finding a particular item may not be easy, and for that the developers introduced a search bar that grays out items that do not match the description by dimming the saturation and brightness of their icons.
4.5.2.2 Examples of Player Exploits

Just as we discussed for Signal Detection, that increasing the graphics settings in World of Warcraft makes seeing certain effects easier, the opposite can be done here.

By reducing the graphics quality in some games, players can get rid of certain graphical elements that make the task of Visual Search harder [Figure 4.13].

![Figure 4.13 PUBG screenshot of the same scene with high (top) and low (bottom) graphics settings, which can unintentionally affect the difficulty of a Visual Search task.](image)

4.5.3 – Assistance Techniques

1- **Highlights**: Highlighting the target to make it stand out.

2- **Visual Connection**: A guide to the target via a visual connection.

3- **Target Details**: Providing additional information about the target.

4- **Compass**: Pointer guiding the player in the target’s direction.

5- **Directional Cues**: An auditory/visual effect to guide the players’ attention.

4.5.3.1 – Highlights

**Definition**

Highlighting the target to make it stand out and become more visually salient, thus reducing the difficulty of locating the target.
Generic Example

The design and implementation of the highlight would be chosen by the game designer. It could be a simple outline around the target, an overlay, or a glow of some sort.

Game Example #1 (Default)

Many of the quests in World of Warcraft ask the player to collect items from the environment. When searching the area of interest for the quest, target items are highlighted with a sparkle effect.

Game Example #2 (Optional)

Once a character is selected in World of Warcraft, players may choose to mark them with a one of the few preset shapes (e.g. square, triangle, or crescent.) [43]. This can help players find a target within a crowd.

Considerations

This technique could potentially make the task a little too easy if the object is immediately identifiable. Therefore, in situations where the visual search is part of the challenge, we can use modified versions of highlighting to help the player but still retain some elements of the original challenge.

Highlighting an approximate area of where the object would be is one way of providing assistance without removing the challenge completely. Another way would be to show the highlight temporarily, or only show the highlight when the player is close enough or a significant amount of time has passed.
It is important to note that this technique is effective only if the object is within the player’s field of view. This is particularly important for games based in 3D environments, or on virtual reality (VR) or augmented reality (AR) systems, since objects of interest are frequently off-screen.

4.5.3.2 – Visual Connection

Definition
Guiding the player to the location of the target by displaying a full visual connection between the player’s current locations (e.g. a cursor on screen, or hand in VR/AR) to the target directly.

Generic Example
An example of this would be a string tied between a character’s hand in a first-person view and the target.

Game Example #1 (Default)
Before the start of an Overwatch match, a line is shown on the ground to guide defending players from their starting location to the first defense point, like in Figure 4.14 below.

Figure 4.14 Floor guide to a defense point in Overwatch.
**Game Example #2 (Optional)**

One of Kunkka’s abilities [58] - *X Marks the Spot* - in Dota 2 [L29] places a marker under a target character and, as that character moves, a line is drawn between the character and the marker [M20]. A trigger then teleports the character back to the marker. Having such a connection helps the player quickly find the marker and the character even in chaotic situations.

**Considerations**

The advantage of this technique is that it provide information on the direction and altitude of the object and points to its exact location when it is within the field of view. In a VR item-search environment, this technique has been found to be the fastest at guiding a player compared to the other visual techniques discussed here [44].

4.5.3.3 – Target Details

**Definition**

Providing the player with additional information about the target for easier identification.

**Generic Example**

If a player is tasked with finding a specific object on a cluttered desk, showing them an image of the target object may help them identify it more quickly than if they are only given a brief description.

**Game Example #1 (Default)**

There is a mission in Grand Theft Auto V that requires the player to snipe a target at a party from a distance [45]. The target is only described and a large portion of the mission is spent on identifying the target using the given details. This would be an example of a
weak implementation, and a stronger one would be giving the player a picture of the target. However, it could be argued that a stronger implementation would have taken away from the challenge and intrigue of the mission.

**Considerations**

A stronger implementation of this technique is one where more information is given.

Images of the target can be more effective than detailed text descriptions since more information is conveyed to the player such as color, shape, size, or type of object.

**4.5.3.4 – Compass**

**Definition**

Implementing a pointer in the game world to guide the player in the direction of the target. Unlike the Visual Connection technique, this one only points the player in a particular direction instead of guiding them directly to it.

**Generic Example**

An example of this could be a compass or an arrow pointing in a specific direction.

**Game Example #1 (Default)**

There are two camera modes in Rocket League [L34]. The first mode, called Ball Cam [M23], locks the camera onto the ball so that no matter how the player moves their car, the ball will be centered in view. The second one centers the camera behind the car so that if the car turns, the camera turns along with it. When Ball Cam is disabled, a small arrow can be seen pointing in the direction of the ball to help players locate it more quickly.
**Game Example #2 (Optional)**

Spyro [L44] is a 3D adventure game where players control a purple dragon and are accompanied by a small firefly that indicates the player’s health and helps with item collection. Each level in the game has a set number of gems that a player would want to collect if they are aiming for 100% completion. It is common for the last gem or two to be hard to find and the player can spend quite a bit of time running back and forth trying to find them. In some of the Spyro games, players are given the option to use the dragonfly to point towards the direction of the nearest gem, making the task much easier.

**Considerations**

This technique only provides the direction as a form of assistance as compared to the full-on guidance of the Connection technique. This technique was found to underperform when compared to the Visual Connection and even the Image of the Target techniques, and it was preferred less [44]. A reason for this may be the fact that with something like an arrow, the player must also pay attention to it and interpret what it means before being able to make use of it.

**4.5.3.5 – Directional Cues**

**Definition**

Cues can be used to guide the player in the correct direction. Depending on the situation, the cue may also be useful to give the player more information about the target itself.

**Generic Example**

An example of this could be following a mechanical humming noise to find out where it is coming from and what is causing it.
**Game Example #1 (Default)**

In certain situations in Dead by Daylight when the killer is winning, players are given an alternate way to escape. Instead of the normal exit gates, players can escape through a *hatch* which only appears after certain requirements have been met. The hatch also requires certain things to happen before it opens, even if it is available.

Players wishing to use the hatch must first find it in the game world. Before the hatch opens, they must rely purely on sight to detect it. Once the hatch is opened, however, a sound effect of rushing air can be heard when the player is near it, which helps guide the player to it when running past it.

**Game Example #2 (Optional)**

One of the perks in Dead by Daylight, *Small Game [46]*, plays a sound effect when the player is looking in the direction of certain intractable objects.
4.6 – Pointing

4.6.1 – Introduction

Pointing refers to the action of moving a pointer towards a specific target [1], specifically paying attention to the speed and accuracy of this action. A simple example of this task would be clicking on an icon on a computer desktop. The pointer is the mouse cursor and the target is the icon.

It is important to note that Pointing is different from Aiming, which is discussed later, in that Pointing includes real-time feedback about the direction of pointing and usually provides immediate response once an action is performed. In the desktop example, the feedback is the position of the cursor on-screen and then once the icon is clicked it would be highlighted instantly. It is important to note that Pointing does not require physical pointing, but usually involves some mapping between an input device and an on-screen indicator (i.e. the pointer).

Pointing in human-computer interaction is extremely well-studied and it is understood that the time taken to select a target is a function of the size of the target and the distance between the pointer and the target [47]. This relationship is referred to as the Index of Difficulty (or ID), where ID is proportional to the distance (D) and inversely proportional to the width of the target (W).
4.6.2 – Examples

Pointing is a common task in games that is usually inherently available due to the use of input devices like a mouse, controller, or even the Kinect. Most pointing tasks may not require any assistance. Such tasks may include menu navigation or making selections in inventory or crafting systems. This is because many of these tasks usually do not have any time constraints and misses are ignored or have no negative effects. This way, even if the player is unskilled with the device (a mouse for instance), they can take their time and not worry about mistakes.

Pointing can become challenging once time and accuracy are taken into account. This is most prominent in games where Pointing is a big part, if not the main part, of the game’s core mechanics. An example of this would be First-Person Shooter (FPS) games where the fundamental gameplay is moving the pointer (in this case usually crosshairs at the center of the screen) towards a target. Doing this too slowly can increase the chances of getting hit first, and missing the target could have some negative implications as well, such as losing ammunition or alerting the opponent to the player’s location.

4.6.2.1 Examples of Pointing Assistance

Because the two main factors that predict pointing performance are the distance to the target and the size of the target, most assistance techniques can be seen as manipulating one or both of these parameters.

In Grand Theft Auto 5, specifically on console, players get to choose to enable “Aim Assist”. This assistance is available in two levels: “Partial” and “Full.” The Partial Aim Assist makes use of Target Locking, which moves the player’s pointer to the center of the screen.
target once targeting is enabled, but after that the player is free to move their crosshairs any way they want (making the effective distance to the target zero). The full version also includes a heavy implementation of Target Gravity, which pulls the pointer back towards the target and can only be overcome by a strong movement away from the target or canceling the action altogether (which reduces both the effective distance and increased the effective size of the target).

Overwatch is another game that provides players with the option for Aim Assistance, but unlike GTA 5, players here can set the ‘strength’ of the assistance on a scale of 0 to 100. Overwatch employs two assistance techniques. The first is Sticky Targeting which slows down the crosshairs when hovering over a target, and the second is Target Gravity which subtly pulls the pointer towards the target whenever the player or the target move. What is interesting is that both techniques are applied at once and it is not possible to enable only one. Overwatch presents it simply as “Aim Assist”.

4.6.2.2 Examples of Player Exploits

Many third party software tools have been created to cheat in games. One of the more common ones found in FPS games is called an Aimbot. Aimbots are usually built as a strong implementation of one of the techniques discussed here. For example, a continuous implementation of Target Lock snaps the pointer to the closest target [M15].

4.6.3 – Assistance Techniques

It has been shown that applying assistance techniques increases a player’s hit ratio [29], which is an improvement in accuracy. If the techniques are applied to only one player in a multiplayer setting, the opposing player would not be affected negatively.
There very well could be more techniques than the ones explored here. To date, there has not been any exhaustive survey of all possible pointing techniques in video games. However, a fairly exhaustive survey has been conducted on techniques for enhancing pointing performance in graphical user interfaces such as PC desktops [47], and many game-based pointing assistance techniques take similar approaches [29][37]. Here, we describe four of the most commonly seen examples.

1- **Sticky Targeting:** Slowing down the pointer when passing over a target.

2- **Target Gravity:** A force pulling the pointer towards the target based on distance.

3- **Target Lock:** Instantly snapping the pointer to the target location.

4- **Area Cursor:** Increasing the area of the pointer (thereby increasing the effective size of the target).

**4.6.3.1 – Sticky Targeting**

*Definition*

Slowing down the movement speed of the pointer when it is on or near the target [29]. The threshold of the effective area can be adjusted to reach the desired result.

This technique can stop players from over shooting their target, especially for players who prefer playing with higher sensitivity settings on their devices. It also helps make small adjustments more easily once the pointer is near the target to increase accuracy even further.
Fitts’s Law [79] is a predictive model of human movement primarily used in Human Computer Interaction (HCI). In terms of Fitts’s Law’s Index of Difficulty, this technique affects the size of the target by a factor proportional to the change in pointer speed.

**Game Example #1 (Default)**

The weapons in Halo 5 use a combination of aim assist techniques, one of which slows down the reticle as it passes over a target [M16].

**Game Example #2 (Optional)**

Along with Target Gravity, this technique is implemented as part of Overwatch’s *Aim Assist* optional feature available on the console version.

**Considerations**

One issue to keep in mind is that once the stickiness kicks in and the pointer is slowed down, the player would have to move their device farther than normal to achieve the same movement distance on screen.

This problem can be ignored if the input is coming from a directional pad (buttons) or a joystick, for instance, since they only have to be pressed/tilted for continuous movement. For certain other devices, however, such as a mouse or a movement-based device like the Wiimote or the Kinect, this may cause some issues. For example, a mouse may go to the edge of the surface it is on before the cursor reaches the edge of the display, or a Wiimote could go beyond the range of its sensor.

This issue called “Cursor Drift” [29] could be overcome by returning the pointer on screen to where it would have been if it had not passed a target and slowed down.
4.6.3.2 – Target Gravity

**Definition**

Giving the target a gravitational force that affects the pointer and pulls it towards the target. Just as it works in reality, the force would be inversely proportional to the distance between the target and the pointer [29]. In terms of Fitts’s Law’s Index of Difficulty, this technique changes the effective distance between the pointer and the target.

**Game Example #1 (Default)**

Fans of Battlefield 4 [L28] have analyzed the game and discovered that it includes a form of assistance to help players aim better [55]. Two techniques were uncovered; one matches the Sticky Targeting technique, discussed earlier, and the other matches Target Gravity. The game code that was examined referred to the implementation of this technique as the “SnapBox”, which seems to be a strong implementation of gravity. These techniques were implemented as part of the game’s default code and is available to all players.

**Game Example #2 (Optional)**

Along with Sticky Targeting, this technique is implemented as part of Overwatch’s *Aim Assist* optional feature available on the console version.

**Considerations**

Different targets could be given different gravity values, which can be based on size or even perhaps importance/priority of the target itself.
One problem that players may face would be getting unintentionally derailed from their chosen target by a second target that may pass across the screen and pull the pointer away from the intended one.

4.6.3.3 – Target Lock

Definition

Instantly moving the pointer to target location. This can be done once, repeatedly, or continuously.

In terms of the Fitts’s Law Index of Difficulty, this technique could be imagined as making the distance between the pointer and target nearly or effectively zero, or perhaps making the size of the target infinite.

Generic Example

An example would be snapping the pointer in a darts game to the center of the dart board.

Game Example #1 (Default)

Aside from the main weapon that players choose to use in Monster Hunter: World, all players have access to a slinger as a secondary weapon. Pressing a specific button switches the player to the slinger mode which brings up a pointer at the center of the screen. Players move this pointer to aim their slinger shot, and they may press a second button to cycle through and lock onto possible targets.

Game Example #2 (Optional)

One of the aiming style options in Grand Theft Auto 5 is called “Traditional GTA” [M17] which is a strong implementation of the target lock technique.
Considerations

The position that the pointer snaps to would be chosen by the game designer. A simple implementation would snap the pointer to the center of the target; however, depending on the context of the game, an offset may be more favorable, such as aiming for the head in FPS games.

The strength of this technique depends on the efficiency of the chosen snap location and the frequency of it happening. For instance, a snap that takes the pointer to the general area of the target once is weaker than one that snaps to the center of the target and continues to follow it accurately as it moves.

4.6.3.4 – Area Cursor

Definition

Increasing the active area of the pointer. The baseline is a single point on screen and increasing in diameter from there. The target would be considered hit even if only partially covered by the pointer.

In terms of Fitts’s Law’s Index of Difficulty, this technique cuts down the distance between the pointer and the target equivalent to roughly the radius of the pointer.

Generic Example

A shotgun’s spread would affect a wider area compared to a shot from a pistol.

Game Example #1 (Default)

Some of the characters in Overwatch, such as D.Va or Reaper, wield weapons that affect
a larger area, making them easier to shoot with, as accuracy is less important. To counter balance the easier pointing task, the developers made those weapons deal less damage.

*Game Example #2 (Optional)*

It is not uncommon for FPS games to give players a choice of weapon, and often times spread weapons are one of those options. In games like Call of Duty or Battlefield, those weapons could be a variation of a shotgun or a wide-spread machine gun. Games that have a fantasy or sci-fi setting could come up with imaginative weapons that can have a similar area effect.

*Considerations*

Even for straight shooting weapons like pistols, this technique could be subtly and unnoticeably implemented by increasing the area effect of the shot without reflecting that visually.

4.6.4 – Pursuit Tracking

Pursuit Tracking is similar to a pointing task, but instead of ending once the target is reached, the player would have to continue to make sure the pointer is on top of the target (or within a target region), especially if it is moving. Since Pursuit Tracking is so similar to Pointing, we consider and describe it here.

Pursuit Tracking is common in many games, for example: lining up a headshot on a moving target in an FPS. In terms of assistance, any of the techniques discussed for Pointing could potentially be extended to help with Pursuit Tracking.
For instance, a continuous Target Lock would essentially take care of this task altogether. Target Gravity can help move towards the target faster or even simply adjust the pointer automatically if the target moves only slightly.

However, some of the techniques may actually be detrimental to Pursuit Tracking even if they were useful for pointing. For example, Sticky Targeting helps with Pointing but it may make tracking a more difficult task. As the pointer would be slowed down whenever it reaches the target, and if the target is moving, it could move faster than the pointer, making it difficult to keep up with it [29]. Furthermore, perhaps a second target would accidentally come between the pointer and the intended target, thus slowing down the pointer.
4.7 – Aiming

4.7.1 – Introduction

Aiming is the task to point at a target [1] without feedback. This is similar to Pointing discussed above, however, Aiming does not include feedback regarding the position/direction while aiming, nor does it necessarily provide an immediate response.

4.7.2 – Examples

Aiming is prominent within First-Person Shooter (FPS) games as it is fundamentally the main gameplay mechanic. However, depending on how the game is presented, the task can vary between being a Pointing task or an Aiming one.

In Call of Duty, for instance, when playing in the easier modes, the player is provided with a crosshair in the center of the screen. This provides accurate feedback as to where the player’s shots would hit. Playing the same game on the harder difficulty setting removes the crosshairs, leaving the player to infer their aiming direction based on the direction of their on-screen gun, which makes the task an Aiming task.

Aiming is encountered more frequently when the action continues to have some sort of movement after being performed, such as throwing grenades in FPS games, or any projectile that follows some rules of physics. The players would have to take into account the fact that their projectiles will continue to move after being shot though the game space, and as such would have to predict their landing spot. They may also need to predict the movement or behavior of a projectile after it is shot or thrown. For example, a grenade may land and bounce, moving from its original landing spot, before it explodes.
Aiming is not exclusive to FPS games however. Fighting and Adventure games can have an element of aiming to them. For instance, in third-person games, the playable character could be rotated to face enemies before attacking or throwing objects. Platformer games use a variety of projectiles that must be aimed at enemies (for example, the fireball in Super Mario Bros.)

4.7.2.1 Examples of Aiming Assistance

Kingdom Hearts [L45] is a third-person adventure game where the player fights multiple targets in a 3D environment, which means that they can attack from any direction. Most abilities in that game are performed right in front of the character (for example, slashing forward with the weapon). This leads to the player to have to constantly turn to face the different enemies.

Players have the option to Lock-On [53] to an enemy which automatically makes the character face the target and continue to do so even if either of the enemy or player move.

Some installments of the game allow the player to unlock enhancements for the Lock-On system such as extending its effective range or automating it to lock on to priority targets without player input.

4.7.3 – Assistance Techniques

The techniques discussed below have been studied as mechanisms to help bridge the gap between players of different skill levels in multiplayer games. It has been observed that if the difference in skill is high, the game might not be as fun for players, experienced
players can be deprived of a challenge, while novices feel that they are not good enough [30].

Just like the techniques discussed for Pointing, these could be analyzed in terms of manipulation of target size and distance which, are a part of Fitts’s Law’s Index of Difficulty.

1- **Target Lock**: Readjusting player position towards the target.

2- **Projectile Magnetism**: Changing the projectile trajectory towards the target.

3- **Area Effect**: Varying the radius of the projectile’s area of effect.

**4.7.3.1 – Target Lock**

**Definition**

Readjusting player position towards the target. Similar to the technique of the same name for Pointing, this could be a one-time adjustment, repeatable, or continuous.

Changing the player direction to face the target may affect the distance in terms of Fitts’s Law, while a continuous implementation of a target lock would effectively increase the size of the target.

**Game Example #1 (Default)**

Modern fighting games have taken advantage of 3D graphics to give players a more realistic experience compared to early 2D platform fighting games, where players could only move on the X and Y axis. However, controlling a character in a 3D environment requires increased challenge. Some 3D fighting games, such as Naruto: Ultimate Ninja Storm [34], provide target locking by default. Players are free to roam around and even
break the lock, but most actions players take would orient them towards their opponent. This way, players would be able to focus on their timing and strategy.

**Game Example #2 (Optional)**

While hunting, players in Monster Hunter: World can press a button to orient the camera towards the target monster. However, by tweaking certain options [M12], players could change it so that the character would also face the target as well for more accurate hits.

**Considerations**

The strength of this technique would depend on the accuracy of the readjustment and its frequency.

**4.7.3.2 – Projectile Magnetism**

**Definition**

Changing the projectile trajectory in the direction of the target. Depending on the rate of change chosen, this technique would help players whose aim may be slightly off-center.

This technique would affect both distance and target size in terms of Fitts’s Law’s Index of Difficulty.

**Generic Example**

An example of this is heat-seeking missiles adjusting their path during flight.

**Game Example #1 (Default)**

Each of the weapons in Halo 5 has a few different settings that are hidden from the player that make it easier to use. One of those hidden settings is bullet magnetism [M16], which pulls the bullet towards the target even if the shot was made slightly off-target.
Game Example #2 (Optional)

Call of Duty features weapon styles, one of which is the launcher type. Launchers are split into two groups, guided and unguided [48]. The guided launchers lock on to a target and shoot out a rocket that will attempt to follow the locked-on target. These weapons are usually more difficult to obtain and sometimes have less ammunition to counterbalance the positive effects of their magnetism.

Considerations

If there are multiple targets near the projectile path, choosing which one to move towards may be difficult. A simple solution would be to turn to the nearest one, but that may not be the player’s intended target, especially if the difference in distance was minimal to begin with.

It would be the designers’ job to think about how to best deal with such situations. For instance, one could choose the most valuable target based on criteria chosen by the designers, such as danger posed by the target or any reward that may be given for hitting that specific target. Another option is to give a gravitational force value to each target so that all targets would exert a pull on the projectile, achieving a net adjustment to its trajectory based on all the forces acting upon it.

4.7.3.3 – Area Effect

Definition

Adjusting the area of effect of a projectile which would vary the chances of the target being within the radius of impact. This technique would also affect both distance and target size in regards to Fitts’s Law.
**Generic Example**

A larger explosion radius for a grenade would make it easier to hit the target, even if the grenade throw was missed as it would still hit the target if it was close enough.

**Game Example #1 (Default)**

Many games provide players with a variety of weapon choices, some of which affect a large area. One such example is Overwatch’s Pharah [33] character who wields a rocket launcher. Each rocket flies through the 3D space and then explodes on impact, dealing damage to any character caught within the explosion radius. This makes it easier for Pharah to hit targets; however, the damage dealt is reduced the further the target is from the center of the explosion to counter balance the easier aiming.

**Game Example #2 (Optional)**

One of the perks available in Call of Duty’s Modern Warfare is *Sonic Boom* [32]. Players may choose to enable this perk, which increases the radius of all explosives the player may use, including grenades and rocket launchers.
4.8 – Steering

4.8.1 – Introduction

Steering is a player’s ability to quickly and accurately move a pointer or an object along a path without going outside the limits of the path [1]. This might seem that it is relevant only to racing games. However, any game task that requires guiding an object (such as a character) through an environment is considered Steering. Steering can be found in many different types of games, including Platform, Adventure, and Puzzle games.

4.8.2 – Examples

Driving is the primary example for Steering. Driving mechanics can be found in racing games such as the Forza franchise, but is also found in other games like Grand Theft Auto, Battlefield, and Farming Simulator [L46].

The controls found in games that feature driving can range from realistic simulations to arcade-feeling controls that can contradict physics. In any of those cases, the task is to move along the path.

Other games can implement Steering mechanics that have nothing to do with driving. In Super Monkey Ball [L47] (3D puzzle platformer), for example, the player controls a monkey inside a plastic ball and may have to navigate narrow platforms or avoid obstacles on a steep slide.
4.8.2.1 Examples of Steering Assistance

Forza Motorsports 6 provides players with multiple assistance options to make driving easier, all of which can be disabled for extra in-game credits. Some of these techniques include showing the optimal path on the road, braking automatically before sharp turns, and even a time-rewind option for players to correct their mistakes.

In Figure 4.15, we can see the arrows on the road showing the ideal path to take. The colors of the arrows also provide useful information. Normally they would be blue, but turn yellow and red when it is suggested to start braking before and during a turn.

4.8.3 – Assistance Techniques

1- **View**: Giving the player more control over their view into the game world.

2- **Steering Adjustment**: Adjusting player velocity towards the optimal path.

3- **Path Guidance**: Guiding the player towards the optimal path.

![Figure 4.15 An example of Path Guidance in Forza Motorsports.](image-url)
4.8.3.1 – View

**Definition**

The way the object and/or scene are presented to players can have an impact on their perception and ultimately on their Steering performance [35]. By providing several view options ranging from full camera control to preset view positions, players can choose one that feels most natural and comfortable to them to improve their performance.

**Generic Example**

In driving situations, designers generally provide one or more of the following views: First-Person (From inside the vehicle), Third-Person (usually behind the car) using different camera angles, Top-Down where the camera is positioned right above the vehicle which can be at different heights, or a bumper-view where the camera would be positioned in front of the car facing forward.

**Game Example #1 (Default)**

Survivors in Dead by Daylight play the game in a third-person view, while the killer is stuck with a first-person view. This makes it easier for the survivors to steer around obstacles and plan their routes as they can see their surroundings more clearly than the killer.

**Game Example #2 (Optional)**

Games that involve driving, such as Forza and GTA, give the player the option to switch between several camera views, such as first-person, third-person, and top-down. Players can switch between them freely and select the one that helps them play best.
Considerations

Some unconventional viewpoints can be created depending on the design and context as well. Grand Theft Auto, for instance, provides a “Cinematic” view, which simulates camera angles as seen in movies, which in this case actually hinders the driving but provides an interesting experience.

4.8.3.2 – Steering Adjustment

Definition

Adjusting player velocity towards the ideal path. The ideal path would be chosen by the game designers. It could be predetermined, dynamically calculated, or simply the center of a lane in a track.

Generic Example

An example of this is an autopilot adjusting the course of an aircraft without user input.

Game Example #1 (Default)

Harry Potter: Quidditch World Cup [L48] is a game based on the fictional sport from the book series of the same name. To end a match, the player must follow the snitch (a flying golden ball) along a path until they are close enough to catch it before their opponent. This is where a strong implementation of this technique can be found, as the player is constantly being pulled towards the center of path [M18].
**Game Example #2 (Optional)**

Certain missions in Grand Theft Auto allow the player to switch between the playable characters. One of the characters may be driving while the other is shooting out the window. If the player is struggling to stay on the right path while driving, they can switch to the shooting character and the driving will be taken care of automatically.

### 4.8.3.3 – Path Guidance

**Definition**

Using cues such as arrows or lines to guide the player towards the optimal path.

Unlike Path Adjustment, however, it would still be the player’s responsibility to move onto that path.

**Generic Example**

An example of this would be the runway lights guiding an airplane for takeoff.

**Game Example #1 (Default)**

When attempting to win a certain medal (Bronze, Silver, or Gold) in TrackMania Turbo [L49], players race against a phantom car of the same color. That car represents the minimum optimal path taken to achieve the target medal.

**Game Example #2 (Optional)**

The Forza example given near the start of this section is a perfect example of this technique. Players may enable a guidance line that shows the suggested path players should follow along the race track.
Considerations

The cues do not necessarily have to be visual. For example, auditory cues can be used to warn the player when they are moving too far from the path, or haptic feedback like controller vibration could be provided when steering off the main path. An example of that would be vibration when the player is driving on the dirt on the outside of a race track.
4.9 – Memory

4.9.1 – Introduction

Players are often required to remember some information in a game, either for immediate use (a short-term memory task) or for future use (a longer-term memory task). This could be the weakness of an enemy, the pin-code for a safe, or ingredients for a crafting recipe. Depending on the scale of the game and the number of items or actions, the pieces of information the player may need to remember can become overwhelming. Having to constantly double check information can get frustrating, and even worse if there is no way to confirm certain details. Players may then have to resort to guessing or searching for the answer online.

4.9.2 – Examples

In some story or roleplaying games such as Life is Strange [L37] or The Wolf Among Us [L38], players are required to make timed decisions as a response to an event. These decisions may have something to do with elements of the story that have already passed, some of which may be based on previous personal decisions.
4.9.2.1 Examples of Memory Assistance

Life is Strange keeps a sort of *Scrap Book* that tells the story up to the point that the player has reached. Players can browse the scrap book to remember what has happened so far and what decisions have already been made by the player, which can be different with every play-through.

4.9.2.2 Examples of Player Exploits

In World of Warcraft, players can put up items for sale on a trading system called the Auction House. Other players can then search for those items and buy them through the Auction House directly.

It is common practice for players to buy crafting ingredients in this way. They would normally have to check the crafting window for the items they need, then close the ingredient list and open the Auction House window, recalling and typing in the ingredient names manually. This can be difficult if several items are required and it can be annoying to have to switch windows often to double check which items are needed.

Players have found that by adjusting settings (such as the screen resolution and UI scaling), it is possible to have both windows open at the same time. This would essentially generate a stronger version of the *Information Archive* (via the crafting window, see below) where the information is more readily available.
4.9.3 – Assistance Techniques

1- **Real-Time Reminders**: Actively reminding the player of information to be recalled.

2- **Information Archive**: A store of relevant information that may need to be recalled.

3- **Announcements**: Highlighting relevant information, reinforcing the fact that they may need to be recalled in the future.

4.9.3.1 – Reminders

**Definition**

Actively reminding the player of relevant information. It would be up to the game designers to decide on what information to remind the player of, including when, where, and how the information is presented.

**Game Example #1 (Default)**

In most of the Pokémon handheld games, specifically those from the main franchise, players are often reminded of certain details when attempting to perform actions with restrictions that are not met, like in Figure 4.17.

![This tree looks like it can be CUT down!](https://i.imgflip.com/gd7ag.jpg)

**Figure 4.17 Ability (Cut) usage reminder in Pokémon [L39] (Fire Red / Leaf Green)**

(Source: https://i.imgflip.com/gd7ag.jpg)

**Game Example #2 (Optional)**

In some games, such reminders can be enabled if the player needs additional help. Sometimes, the reminders can be triggered when needed rather than automatically like the one from the previous example. One such example would be the hint system often found in “Find the hidden object” games.
Golden Trails [L27] provides players with a hint button (seen in Figure 4.18 below) that they may choose to press and reveal relevant information that is often a reminder to something the player may have forgotten but needs to know at the time.

![Figure 4.18 An example of a Reminder: hints in Golden Trails](https://drawinglics.com/view/4912251/golden-trails-the-new-western-rush-hidden-object-game-hint-scene-in-game-golden-trails-the-new-western-rush.jpg)

**Considerations**

Information can sometimes completely remove the challenge for a player. Therefore, limiting the use of reminders, especially when they are optional like the one in Game Example #2, can help reduce the strength of this technique. Making this information limited can also add a new layer of challenge and strategy to the game as players have to make wise decisions as to when to rely on these reminders.
4.9.3.2 – Information Archive

**Definition**

Providing an archive of information relevant to the game for the player to explore. It would be the designers’ choice to decide what information to keep track of, for instance whether to only record information that the player has already discovered. The designers must also decide when and how the information can be accessed, through the menu for example, or by speaking to a character in game.

**Game Example #1 (Default)**

The Witcher 3 has a Bestiary Guide [49] that keeps track of all the monsters and creatures the player has encountered in the game so far. It includes information such as monster descriptions, and weaknesses.

**Considerations**

The strength of an implementation of an information archive depends on content and accessibility.

The value of an archive’s content depends of its accuracy, detail, and quantity. An archive describing game items in detail would be considered stronger than one simply listing the item names.

As for accessibility, an archive viewable while performing a related action would be considered stronger than one only viewable during player downtime. For example, in World of Warcraft as described above, players find it much easier to have the reference
ingredients available while using the Auction House, rather than having to close the Auction House window and reference the ingredients separately.

4.9.3.3 – Announcements

**Definition**

Highlighting important information for the player that they may need to recall in the future. This works by directing the player’s attention to useful information worth memorizing, effectively reducing the amount of information the player would otherwise attempt to keep in memory for later.

**Generic Example**

An example would be telling a player that a sequence of numbers they had just uncovered will be relevant at some point later on in the game (to unlock a door for instance).

**Game Example #1 (Default)**

Many of the non-player characters (NPCs) in the Witcher 3 give the player clues and tips on how to kill certain monsters as part of their in-game dialogue. Players who pay attention to such information may have an advantage when encountering those monsters. The same information can still be learned through trial and error, the many in-game books, or the in-game bestiary, which is an implementation of an Information Archive discussed earlier.

**Game Example #2 (Optional)**

Telltale’s story-driven games such as the Walking Dead series, or the Wolf Among Us, often announce to the player when an action or decision may be important for later. These
announcements take the form of short messages such as “You may need this item later” or “This person will remember what you did.”

These announcements are enabled by default, but players can choose to disable them to increase immersion and difficulty as they would then need to guess whether certain information is important.
4.10 – Spatial Memory

4.10.1 – Introduction

In terms of psychology, spatial memory refers to the part of memory responsible for recording information about the environment and spatial orientation [50]. Spatial memory tasks, therefore, require the recall of a specific location. Just like memory discussed in the previous section, spatial memory could be short or long term. In games, this translates to the player’s ability to recall the locations of game elements in the virtual environment, or to accurately visualize their orientation within the game world. Since spatial memory tasks are so common and the techniques that support it are different from Memory tasks, we consider them separately.

4.10.2 – Examples

In open world games like World of Warcraft, Grand Theft Auto, or Slime Rancher [L50], players often have to remember where a certain in-game shop is, such as where a specific creature lives, or where events happen.

Figure 4.19 (Left) Dialogue with city guards in World of Warcraft listing commonly visited points of interest. (Right) Once a player selects a point of interest option, that location is marked on the in-game world map. This demonstrates a use of the Maps technique to assist with Spatial Memory.
4.10.2.1 Examples of Spatial Memory Assistance

When searching for certain things in World of Warcraft, players can choose to enable additional map icons that highlight the locations of vendors, mailboxes, banks, etc.

Players may also speak to city guards in-game and ask for directions (Figure 4.19 (Left)) which are then visualized by an icon on their world map (Figure 4.19 (Right)).

4.10.3 – Assistance Techniques

1- Maps: A visual representation of the game environment.

2- Markers: Markers in the game environment to inform the player of a location.

4.10.3.1 – Maps

Definition

Providing the player with a map of the game environment. This may take the form of a world map showing the locations of cities and such, regardless of where the player may be, or a mini-map that gives the player information about their immediate surroundings updated in real time.

It would be up to the designer to decide how much information the map conveys, and how much of it is available to the player at a given moment. Some games only show areas the player has already visited. Others provide additional information on the map for players with specific in-game abilities, for example, the ability to track where a certain type of item can be found.
Game Example #1 (Default)

Open world roleplaying games such as The Witcher 3 (Figure 4.20) or Grand Theft Auto 5 (Figure 4.21) provide players with a world map and a mini-map as described above.

Game Example #2 (Optional)

Nearly all of the Call of Duty games give players the option to play on “Hardcore Mode” [56], which is the highest level of difficulty available. One of the features of this mode is the absence of a mini-map.

Figure 4.20 Witcher 3’s world map is an example of the Maps technique. (Source: https://guides.gamepressure.com/static/mapy/en/gfx/map_1236.jpg)

Figure 4.21 An example of Maps from a different game, Grand Theft Auto 5 (Source: https://img.gta5-mods.com/q95/images/singleplay-reveal-map/43dcff-633-55fd51b57c5c7.jpg)
4.10.3.2 – Markers

Definition

Markers placed in the game environment to guide the player to a specific location. These markers are often implemented using billboarding techniques [51], and usually visible through obstacles.

Game Example #1 (Default)

When playing a healing character in Overwatch, the locations of friendly team members are marked so that the healer can find them more easily. These are billboard style markers that are visible through all in-game obstacles if the target character is not within the player’s line of sight.

Game Example #2 (Optional)

Once a character is selected in World of Warcraft, players may choose to mark them with a one of the few preset shapes (square, triangle, or crescent) [52]. These shapes are billboard style markers that appear above the head of the selected character and are also visible through in-game obstacles. Players can use these markers to help them return to a particular character, especially if that character can wander off.

Considerations

Some implementations of markers require the player to be looking in their direction; others may give the player an indication near the edges of the display to indicate that the player may need to turn.

Markers do not necessarily need to be visual. For example, a continuous directional sound effect may be used to guide the player to a particular in-game location.
4.11 – Assistance Technique Selection & Calibration

Because there is a large number of techniques to consider, we discuss the process of the appropriate selection of a technique and its calibration in a particular game to reach the desired difficulty level.

Step 1: Core Task Identification

The first step is to identify the core tasks within the game. We touched on this briefly in the introduction to assistance techniques, with Figure 3.1, where we identified the core tasks from Super Mario Bros.

Firstly, we make a list of high level player actions by answering questions like “What can the player do in this game?” Some possible answers may be “Shooting a gun”, “Collecting items”, “Running around”, etc.

Then we examine each of the high level actions in more detail and try to describe them directly through associating them with one or more core tasks. For example, “Shooting a gun” requires the player to point and click, “Collecting items” may include the player moving (Steering) to the item location (Spatial Memory) and then finding the item within a messy environment (Visual Search).

Step 2: Assistance Goals

To choose the correct technique, we would first need to know what our goal is. In this section we can discuss questions like “Why are we providing assistance?” and “How strong do we want it to be?”
We have now identified the core tasks that can be assisted in the game. To implement the appropriate assistance technique, we must first have a clear goal in mind. We should be able to answer questions such as “Does this task need assisting?” “Does it need to be more or less difficult?” and “What aspect of this action needs assisting?” For that last one, we are referring to player actions that consist of several core tasks like a charged ranged weapon that may require aiming and reaction time.

For example, if collecting items is the main challenge of the game, like in Animal Crossing [L23], then a weaker implementation (or no assistance at all) for tasks like visual search or reaction time might be chosen. However, if the item collection is a part of a looting system after a difficult boss encounter, such as the monsters from Monster Hunter: World, then a stronger implementation would make sense for visual search since the players have beaten the challenge and picking up the reward should not be difficult.

**Step 3: Selecting Techniques**

Each of the core tasks identified by this point could have several potential assistance techniques to be implemented, many of which may be suitable to reach the desired goals from Step 2, however, there are more things to be considered when selecting one of them.

**Design and Theme**

Some techniques may be harder to implement than others, based on the type of game being built, as there can be a fundamental mismatch between a game theme and a particular assistance technique. For instance, a basketball game could implement some of
the *aiming* techniques to help players score. Target Lock would make sense by turning the player character towards the hoop (target). However, something like Projectile Magnetism may look odd if the ball turns in mid-air.

Even then, the technique can still work, it would just *feel* out of place. However, designers of a fantasy or sci-fi themed basketball game could give their characters supernatural powers that would explain the magnetism of the ball.

**Presentation**

Once we select a technique, we need to decide on how it would be delivered to the player.

**Default:** The technique can be implemented as part of the game and made available to all players at all times. For example, *visual search* highlights are used in Monster Hunter: World and are available to players by default, as finding and picking up items is not the main challenge of the game.

**Optional:** The technique would be available to players who *choose* to enable it, or in some cases, it would be enabled by default and disabled by more experienced players. The way such an option is presented can be as simple as a menu toggle or can be creative, using parts of the game like character abilities, items, and power-ups.

**Dynamic:** This is a mix of default and optional applications of the technique, where the assistance is given to the player dynamically. It would be up to the designers to find the best way to implement such a system. This could be as simple as providing assistance
after a certain number of failed attempts or as complex as implementing an AI system to
analyze player performance and make tailored assistance decisions.

In multiplayer games, designers would also have to decide on who receives assistance,
and when and why they receive it.

**Step 4: Calibration**

Once an assistance technique is selected, we need to discuss its *calibration*, which refers
to the strength of its implementation. Strength is discussed in more detail as part of the
assistance techniques introduction section earlier (see 3.2.5 - *Strength of Assistance
Techniques*). For many of the techniques, the strength is self-explanatory (such as Target
Gravity). However, for the ones that are not, a brief discussion on strength would be
found in its *considerations* section.

Calibration is important because simply implementing a technique may not be enough to
reach the desired goal, or perhaps it could be too much if a specific degree of challenge
and difficulty is still desired. To fine tune the implementation of a technique, playtesting
can be done to see how it affects player performance.

In Chapter 5, we demonstrate this process through the application of several techniques to
three different games.
5 – Demonstration and Evaluation

5.1 - Demonstration

To show that the descriptive framework has value, we demonstrate how we can adjust these techniques to match different genres. We developed three different games that share a core task – Visual Search. We then implemented three techniques to each of the games: Highlights, Target Details, and Compass.

Visual search was selected for the demonstration because it has not been closely examined academically in terms of video games, especially compared to Pointing [29], Aiming [30], or Steering [35]. As for the techniques, we chose to do more than one to help highlight their diversity.

We wanted to show that it is possible to successfully implement a technique into different games. One game would not have been sufficient and therefore we decided to build three games and make them as different from each other as possible. However, we believe this would have worked with more examples just as easily.

The first game seen in Figure 5.1 (Left) is a simple top-down view with target selection. A straightforward visual search task here is similar to what can be found in puzzle or point-and-click games. Figure 5.1 (Bottom) shows a third-person game with the targets spread all around, which is similar to adventure and roleplaying games. The final game is a sniper simulation shown in Figure 5.1 (Right). Using snipers is popular in first-person shooting games and some roleplaying games like Grand Theft Auto.
The games are not only differentiated by style and view, but also by their core tasks. The first one (Figure 5.1 (Left)) is mainly a visual search game but it does include pointing. The second one (Figure 5.1 (Bottom)) has elements of steering for moving the character and spatial memory for remembering where a player has already looked, which is shared with the sniper game. The sniper game also has an element of signal discrimination, as some of the environmental objects may look like a target from that view.

In each game, the target is given to the player in simple text – near the bottom of the screen. Their score in the top right corner. A correct click grants the player two points, while clicking on any other shape removes one point.

The targets used in all three games are simple shapes such as cubes, spheres, stars, and hearts. However, in a “real” game, these can take the form of anything that may be
relevant to the game. For example, Figure 5.1 (Left) can be the candy pieces in Candy Crush Saga [L24] as seen in Figure 5.2 (Left). For the third-person view (Figure 5.1 (Bottom)), the targets could be some in-game loot such as weapons or gold like those from Figure 5.2 (Bottom) - Diablo 3 [L25]. Snipers are usually used to hit moving targets – enemies – such as in Sniper Elite [L26] shown in Figure 5.2 (Right).

Figure 5.2 (Left) Candy Crush Saga. (Bottom) Diablo 3 - Loot. (Right) Sniper Elite - Sniper View

Sources:
https://d4c2us8g123wy.cloudfront.net/screens/candycrush.png
http://www.gamersdecide.com/sites/default/files/authors/u138773/loot.jpg
http://www.gamershell.com/static/screenshots/0/135/187351_full.jpg
5.2 – Evaluation

To evaluate the effectiveness of assistance techniques that were created based on our framework, we ran a study using two of the three games: Top-down view (Figure 5.1 (Left)), and third-person (Figure 5.1 (Bottom)). These two games were chosen as they are most distinct from one another, and it allowed us to keep the duration of each session manageable, while still being able to test all of the implemented techniques.

5.2.1 - Apparatus and Setup

The games were designed and built using the Unity 3D [59] game engine. They were developed entirely in C# [60]. Game sessions were played on 64-bit Windows 10 machines, Intel Core i7 CPU and a DELL 19-inch, 1440 x 900 monitor. Data logging was done through an in-game C# script that wrote to an identifiable comma-separated values (CSV) spreadsheet file [61].

Data was logged in two files; the first one included data for each successful trial in the experiment (i.e. a trial required finding and clicking a target) and the second summarized the players’ overall performance for each of the two games. Both files shared common information such as the player ID, type of game being played, and which assistance technique was enabled (including “none”).

The summary log was comprised of the players’ final scores for each game, the number of targets found, and the mean time between each successful click, as seen in Table B-2 in the Appendix. Meanwhile, the per-trial report includes many more entries, one for each successful click showing the name and color of the target, the number of clicks before it was found (minimum one), the time from when that target was announced to
when it was found, and the coordinates and distance of the new target relative to the previous one. An example of this can again be found in Appendix B - Table B-3.

5.2.2 - Participants

We recruited 16 participants to play the games, each of whom were compensated with a $10 gift card. Participants ranged in age from 18 to 43 years old (M=25.375, SD=6.889), 4 identified as female and 12 identified as male. 75% of the participants were university students, ranging from undergraduate to PhD candidates.

The participants had a wide range of gaming experience, but most participants have played games using a mouse and keyboard. A majority of the participants preferred the WASD [62] (62.5%) input scheme over the arrow keys (12.5%) and the remainder had no preference. One of the two games that the participants had to play used keyboard and mouse input to control the in-game character. Participants that were not familiar with the WASD input scheme spent more time in Practice Mode for that game to get comfortable with the controls since it was more convenient than the arrow keys.

5.2.3 - Experimental Task

For each run of the game, participants were given two minutes to locate and click on a target amongst a field of distractors. On a successful click, all the shapes in the game scene are randomly regenerated (to ensure that the game required visual search and not spatial search), and then a new target is presented to the player.

While playing the top-down version of the game, players are only required to use the mouse to move the on-screen pointer to the target as all shapes are visible at once. During
the third-person version, players had to use the keyboard to move their in-game character around to look and search, as not all shapes can be visible at once.

The keyboard controls follow the standard WASD scheme [62], with which most of the participants were familiar. All the participants used the WASD input scheme, even those who stated that they prefer arrow keys, since it was more convenient, especially with mouse input required simultaneously. Participants were given the chance to practice before each trial, and those who were unfamiliar with the WASD input scheme spent as much time as they needed in practice mode to become comfortable with it.

5.2.4 - Techniques

In the Demonstration section (see 5 – Demonstration), we briefly mentioned that the three techniques implemented in each of the games are Highlights, Hints, and Compass. This section will delve a little further into the implementation of each one highlighting any deviations in implementation between games.

5.2.4.1 - Highlights

For the top-down game, highlights were generated as a larger, transparent, copy of the target, as seen in Figure 5.3 (Left). This creates a sort of ‘glow’ around the target, which is common in many games to highlight that an object is able to be interacted with [65].

As for the third-person version of the game, Highlights was redesigned to be stronger, with the players’ limited view in mind. The Highlights implementation here is a beam of colored particles shooting upwards from the target (Figure 5.3 (Right)). These are similar to the colored markers found in World of Warcraft [52] or Fortnite [L30][63].
The variation in the implementation of Highlights for these two games is further explored in Chapter 6 - Discussion and is related to section 4.11 – Assistance Technique Selection & Calibration presented earlier.

5.2.4.2 - Hints

This was an implementation of Target Details which gives the player more information about their target. Normally, only the name and color are given to the player in form of text. With this technique enabled, players are shown an image of their target, which helps them avoid having to visualize the target and make quicker comparisons with what they see on screen. The implementation for both games is nearly identical, except perhaps for the placement of the hint image on screen, as seen in Figure 5.4.
5.2.4.3 - Compass

A 3D pointer was implemented into the scene that is rotated to point towards the next target. The rotation was updated in real-time and moved once a new target was generated, the current target was moved, or the player moved (mainly in the third-person view).

The compass was placed at the bottom-center of the screen in the top-down version of the game, as seen in Figure 5.5 (Left), and under the playable character in the third-person view, as shown in Figure 5.5 (Right).

![Figure 5.5 Compass technique, displayed in the Top-Down game (left) and Third-Person game (right).](image)

5.2.5 - Procedure

The evaluation experiment was designed to be completed in less than 60 minutes. The game consisted of completing the experimental task 8 times, once for every combination of the independent variables: 4 technique levels (no assistance, compass, highlight and hint) and 2 game type levels (top-down and third-person). The gameplay portion of the experiment required approximately 16 minutes. Before beginning a play-through, participants were given a brief introduction to each new technique and new game, and provided an opportunity to get accustomed to the combination by playing the game in a practice mode. Participants were informed that they could take as much time as necessary...
to get comfortable with a specific technique in a specific game. Overall, training required approximately 10 minutes per participant.

The remainder of the time was taken up by the questionnaires, as described in the Participants section, and a few minutes were needed at the start of the experiments for an introduction, the signing of an informed consent form, and at least one practice play session until the participant was comfortable and ready.

The participants were asked to fill out several questionnaires throughout their session. One questionnaire was presented at the start of the experiment session to collect demographic data, including gaming experience, specifically with a mouse and keyboard (see Appendix – Figure B-1 Questionnaire: Pre-Experiment). After each play-through, participants were asked to fill out a brief questionnaire about their experience with the most recent game/technique they played (see Appendix B – Figure B-2 Questionnaire: Post Technique).

A similar third questionnaire was filled out after all the techniques for one game had been played through (see Appendix B – and Figure B-2 Questionnaire: Post Technique). A final questionnaire was presented once participants had completed all the games. This one included a few open-ended questions for the participants to share their views and opinions on assistance techniques (see Appendix B – Figure B-3 Questionnaire: Post Game).

The presentation of techniques was balanced between participants by rotating the sequence of techniques played. The first participant played each (Top-Down always first)
game starting with no technique enabled followed by Highlights, then Hint, and finally the Compass technique. The second participant started with Highlights and ended with None, the third began with Hint and ended with Highlights, and so on.

5.2.6 - Results

5.2.6.1 - Player Performance

We examined two metrics to gauge player performance, the mean time between successful clicks, and the final score achieved by the player. Both were collected for each game/technique combination, as described earlier in section 5.2.1 - Apparatus and Setup.

For the mean time, a lower number is considered better since the players are finding their target more quickly as we can see in Figure 5.7. The chart shows the mean time per successful click while playing one of the games with one of the assistance techniques enabled or none at all.

As for the player score, naturally a higher number would indicate better performance. Players were asked to try their best and achieve the highest score possible within the two minutes of gameplay. We expected that the proportions of these values would be inversely proportional to the mean time and we can see this holds in the mean score results visualized in Figure 5.6 below.

Participants did provide a consistent level of effort throughout the experiment. On several occasions, we observed participants players racing against the clock to hit one more target, and they would be visibly upset if they were close enough to their final target just as their time ran out.
5.2.6.2 - Statistical Analysis of Performance Results

We used a Two-Way ANOVA analysis with within-subject factors (technique and game type) to analyze the data collected during the study. This section highlights the results of the analysis and compares it with the observations above.

Score

![Bar chart showing mean scores for different game and technique combinations](image)

**Figure 5.6 Mean score for each game and technique combination played.**

The overall mean score for all games was 66.0 points (SD=50.467). The Top-Down game had the higher mean score of 102.1 (SD=46.698), while Third-Person had the lower mean score at 30.0 points (SD=17.669). The main effect of Game Type on player score was statistically significant \(F_{1, 15} = 235.442, p < .01\).

As for the effects of Assistance Techniques on player score, we found that the highest mean score was achieved with Highlights enabled, at 106.6 points, followed by Compass with 68.0 points, and finally Hint and No Assistance came near the bottom with 49.2 and 42.1 points respectively. The main effect of assistance techniques was also significant \(F_{3, 45} = 107.166, p < .01\).
There was a significant Game Type x Assistance Technique interaction effect \( (F_{3, 45} = 50.680, p < .01) \), which was due mainly to the significant differences between Highlight and the other techniques for the Top-Down game and nearly all the pairs for Third-Person (except for None/Hint and Highlight/Compass, which were not significant), as determined by a Scheffé post hoc analysis.

This matches our observation as Highlights easily overshadowed the effects of the other techniques in the Top-Down game, while in Third-Person, Hint was not much better than No Assistance, while Highlights and Compass helped overcome the difficulties of the third-person perspective better. Both effects can be seen clearly in the score charts shown in Figure 5.6.

**Time**

![Figure 5.7 Mean time (seconds) per successful click (Target Found) for both games.](image)

The overall mean time per successful click for all games was 5.70 seconds (SD=5.253). The Top-Down game had the faster mean time of 2.70 seconds (SD=1.818) per successful click, while Third-Person had the slower time at 8.60 seconds (SD=5.870) per
successful click. The main effect of Game Type on the mean time per successful click was statistically significant ($F_{1, 15} = 69.015, p < .01$).

As for the effects of Assistance Techniques on the player time per successful click, we found that the quickest mean time was achieved with Highlights enabled at 3.38 seconds, followed by Compass with 5.24 seconds, then Hint at 6.85 seconds, and finally No Assistance had a mean of 7.40 seconds per click. Those effects are also statistically significant ($F_{3, 45} = 6.347, p < .05$).

This time, however, we found that the Game Type x Assistance Technique interaction effect was not statistically significant ($F_{3, 45} = 1.678, p = 0.1853$). This could indicate that the type of game did not have a significant impact on the effect of the assistance technique. This can be further demonstrated by the similar trends seen in Figure 5.7, even though the range of values (y-axis) differs for both.

5.2.6.3 - Player Opinions and Perception

Participants filled out several questionnaires throughout the experiment, as described above.

**Questionnaire Results – Post Technique**

The charts in Figure 5.8 and Figure 5.9 show the average answers for the questions asked after each play-through. The NASA Task Load Index (TLX) [80] answers were aggregated into one value representing “Task Loading” (a lower value is better).

The other three are mean values of agreement with the following statements: “I would like to see similar assistance techniques implemented in video games”, “I have seen a
similar technique in games before”, “I had fun playing this game”, and “I am pretty skilled at this game.”

Figure 5.8 Player agreement ratings for Top-Down game by Assistance Technique

(0 = strongly disagree, 4 = neutral, 7 = strongly agree)

Figure 5.9 Player agreement ratings for Third-Person game by Assistance Techniques

(0 = strongly disagree, 4 = neutral, 7 = strongly agree)
Questionnaire Results – Post Game

Figure 5.10 shows the overall player opinion for each game in regard to these statements:

“The game was easier with assistance techniques”, “I preferred playing with an assistance technique enabled”, and “I feel that such techniques can be useful in similar games.”

On average, our participants agreed with all three statements, finding that the games were easier with assistance enabled, that assistance can be useful and that they would prefer to play with assistance enabled if given the choice.

We analyzed the data collected for each of the questions using a Wilcoxon Signed-Rank test and found all three questions were not statistically significant (p > .05) between the two games played. This means that the participant’s preferences reflected in the chart below (Figure 5.10) are irrespective of the game and the statements from the previous paragraph on the average participant agreements hold regardless of the game played.

![Figure 5.10 Player agreement ratings for assistance effectiveness and usefulness](image)

(0 = strongly disagree, 4 = neutral, 7 = strongly agree)
We then asked the participants to rate the techniques in terms of how effective they believed they were. The question players were presented with was “Rank the techniques in terms of their ability to help your performance. (1 = Most helpful)”. Figure 5.11 below shows the *inverted* mean results of these ratings, where a higher number indicates a better rating from our participants.

For this analysis, we conducted a Friedman test on each set (per game) finding that both were statistically significant: Top-down ($\chi^2 = 30.042, p < .01, df = 3$) and Third-Person ($\chi^2 = 22.350, p < .01, df = 3$). A post-hoc pairwise comparison for both shows that for Top-Down all the pairs were significant except Hint:None, while for Third-person two pairs were not significant (Highlight:Compass and Hint:None, respectively).

![Figure 5.11 Assistance Technique Rating – Top-Down (Left) Third-Person (Right) (4=best, 1=worst)](image)

Finally, players were asked to *choose* the technique they felt was best for the game they had just finished playing. The question presented was “Which technique did you feel best fit with this type of game?” The pie charts in Figure 5.12 below illustrate the distribution of our participants’ choices.
Questionnaire Results – End of Experiment

At the end of the experiment, after playing all of the games, participants were presented with a final questionnaire. Unlike the previous questionnaires, this one was open ended and participants were instructed to answer as they saw fit. Some participants gave longer answers than others, but all participants answered all the questions presented. In this section we summarize the answers collected for each of the questions, highlighting interesting quotes from our participants.

**Question 1: In what type of game, or game scenario, would you imagine such techniques implemented?**

The answers given to this question were varied. However, they seemed to have a common theme where the participants imagine such techniques in “open world” or “3D” games. Participants referenced games such as Grand Theft Auto, World of Warcraft, and Legend of Zelda: Breath of Wild [L31] to confirm the open world pattern. Others pointed to racing games, strategy games and shooting games such as Fortnite.
This makes it seem that the participants were imagining games with environments that encourage exploration and elicit a visual search task in one way or another.

**Question 2: Would you like to see such techniques implemented in games wherever appropriate?**

This question was straightforward with all participants answering “yes”. 75% of the participants simply agreed by saying “yes” or by providing a short explanation as to why they feel that way, such as p1 that said “some games have very good potential but discourages player by not offering help to proceed when the player is stuck”.

The remainder of participants also agreed, but gave their approval of assistance with some conditions such as “a toggle menu allowing players to turn on or off the techniques would be beneficial” for p15, while p11 said “I would like to see it applied in navigation systems of online games.”

**Question 3: If assistance is being used in a game you are playing, would you like to be told about it?**

Just like question 2, this one was quite straightforward. The majority of the participants (75%) said “yes”. The remainder of participants said no, with one of the participants (p12) stating “I think that if a game told me about all the stuff it's doing for me that it could get really annoying quickly. After all, a good interface should be near ubiquitous.” Another participant (p15) pointed out that “Knowing that it was there and performing poorly would not be beneficial as you would feel as though even with assistance, you could not perform well in game.”
**Question 4:** Would you like assistance to be applied automatically? Or would you prefer an option to enable/disable it yourself?

The participants unanimously agreed that having the option to enable and disable assistance is preferable.

Some notable quotes from our participants included p1 saying “it should be easy to access while in game” and participant p5 saying “I may need it more in some games and not at all in others, and it may be pointless for me to be using it if it's a game where I don't need it.”

**Question 5: Comments**

Participants were given the opportunity to voice any comments or concerns they may have had about the experiment as a whole.

Most of the comments made were about the games themselves. Some participants mentioned how much fun they had, others pointed out that the third-person game was more difficult, and a few participants either expanded on one of their answers from previous questions or simply repeated themselves.

Some of the standout comments, however, were complaints about the similarity between certain colors, pink and purple specifically, which could indicate a Signal Discrimination difficulty where a technique such as High Contrast might be useful. Others pointed out the Highlights made the game too easy, which is a comment we heard in person as the justification for choosing Compass over Highlights when asked to choose their favorite.
5.2.7 - Summary of Results

1. Participants did not perform as well in the Third-Person game as the Top-Down game.

2. Highlights and Compass performed similarly in the Third-Person game.

3. Hint (Target Details) was no better than No Assistance in the Third-Person game.

4. Highlights performed significantly better than other techniques in the Top-Down game.

5. Participants found the techniques useful and would prefer to use them when possible.

6. Participants rated Highlights as the best technique for the Top-Down game.

7. Participants rated Highlights and Compass highest for the Third-Person game.

8. A majority of the participants would prefer to know if assistance is provided in a game.

9. Participants prefer to have the ability to toggle assistance on and off in a game.
6 - Discussion

6.1 - Explanation of Results

6.1.1 - Why does Third-Person have a higher mean time and lower mean score than Top-Down?

Looking at the charts (Figure 5.7) from the player performance results of the evaluation (see 5.2.6.1 - Player Performance), we see that the mean time per successful click in the Third-Person game was more than double that of the Top-Down game for each assistance implementation. This difference was mainly due to the type of game being played, which is further corroborated by our statistical analysis of performance results.

Participants had full view of all possible targets during the Top-Down game. While playing the Third-Person version, however, players had a constricted view of the game world with only a subset of targets visible at once. Players had to determine that their target was not in view first, then adjust their view by moving the in-game character and camera to search through the remaining targets. The repeated view adjustments were a major factor in the increased mean time, especially for the participants with limited PC (Keyboard and mouse) gaming experience.

This also led to the large difference in mean score between the two games, which is reflected in the charts of Figure 5.6. Nearly all participants were conscious of the fact that their scores were lower in the Third-Person game, to the point that many of them expressed their concerns to the experimenter about not being able to score as high as Top-Down during the experiment.
6.1.2 – Did some techniques not perform differently in the Third-Person game?

After performing the statistical analysis on player mean score, the Scheffé post hoc analysis revealed that two pairs of techniques did not have a significant difference. These were Highlights and Compass, and the Hint (Target Details) and No Assistance pair.

As we described above, (see 6.1 - Explanation of Results), one of the main factors for lower scores and higher timings in the Third-Person game was the time required to adjust the game view and physically navigate through the game world until the target was located. Both Highlights and Compass help narrow down the player’s search area. We originally expected Highlights to outperform Compass by a substantial amount. However, it seems that environmental obstacles and the distance to the target led to a nearly identical effect between the two. It is important to evaluate such assumptions as it might not be enough to make educated guesses about which techniques are more effective.

Similarly, the increased detail about the target (with the Hint) did not make finding the target faster, if the target was not on screen to begin with. With the compass, players would have to first determine whether the target was onscreen or not, and then readjust their view, which was still just as time-consuming as if there was no assistance at all.
6.1.3 - Why did Highlights lead to the highest scores and lowest times?

Highlights was by far the most effective at increasing player score and reducing the mean time per successful click in both games, although in Third-Person, Compass was close in performance in terms of mean score. These effects can be observed in the mean score and timings charts found in Figure 5.7 and Figure 5.6.

This was not necessarily a surprise, since we deliberately ensured that we had a strong implementation of the Highlights technique in both games. That is one of the reasons it had to be different in Third-Person. More details about that change are discussed below.

From our observations during the experiment and from participant comments, we found that the reason Highlights was so effective is because it makes the target stand out immediately. In both games, Highlights makes the visual saliency of the target to the point that the target has a “visual popout” [75] effect that allows players to identify a target without performing a visual search task. When this happens, players no longer need to spend time looking for their target and instead follow the next highlight to wherever it is. Several participants referenced this fact in some of their questionnaire answers with participant #1 stating that “Highlight was very helpful, but I didn't even have to read the description [of the target] thus removing the essence of the game.” Others also stated that “Highlight helps a lot.” We also believe that this is the reason players often chose Compass over Highlights when asked to choose their favorite in the post-game questionnaire. We discuss this in more detail below.
6.1.4 - Why was Compass the second best in terms of performance?

If we look at the mean scores with each technique by game type (see Figure 5.6), we notice that Compass is second best after Highlights for both games, more so in Third-Person where its mean score was not significantly different from Highlights.

We believe Compass was effective because it narrowed down the area for visual search tasks. For instance, in the Top-Down game, if the Compass arrow points right, players can safely assume that the target is not on the left and immediately discard all targets on the left half of the screen from the list of potential targets. We even observed several participants moving their mouse cursor to the Compass every time a new target was introduced and following the invisible line in the direction the Compass was pointing to find their target.

This effect was stronger in the Third-Person game as view adjustment was a more important factor, because it is possible for the target to not be on the screen at all. Having the Compass quickly tell a player that the target is not on screen, saves them valuable time that they would otherwise have spent checking all the targets on screen before making that same decision.
6.1.5 - Why was Hint no better than No Assistance?

In both games, Hint (Target Details) had the least impact on mean player score and the mean time between successful clicks. Nevertheless, it can still have value for designers, perhaps providing players with the feeling of assistance without necessarily impacting performance. It may also be more effective in situations where the targets are complex.

We believe that the reason it had such a small impact in our games is that after the player is aware of their target, they still need to search for it and at that point it makes no difference to the player whether a hint is given or not.

We did observe, however, that several participants had made mistakes interpreting the characteristics of their next target when presented with a text description only. For instance, we often heard remarks such as “I forgot what an octagon looks like” (p5) after accidentally clicking on a pentagon. Others said things like “Oh, I was searching for a white cube instead of green!” (p11) after clicking on an incorrect target.

6.1.6 - Why did participants prefer Compass almost as much as Highlights?

Our assumption before starting the experiment, and even after running a few pilot studies, was that most participants would choose Highlights as their favorite technique. We made this assumption since we believed it would be the technique to have the largest effect on improving performance – which it did. However, we were surprised to see that Compass was chosen as the most preferred technique nearly as much as Highlights.

We had two questions that relate to assistance technique preference. One was fairly straightforward where the participants were presented with a list of techniques (including
“None”) and asked to choose their favorites. The pie charts in Figure 5.12 show the distribution of participant choices. For Top-Down, all three techniques have been chosen at least once, but Highlights and Compass were most preferred. In fact, Compass was selected as most preferred by one additional participant over Highlights. In Third-Person, Highlights and Compass were the only two chosen as favorites, both selected most preferred equally.

The second question related to this matter had the participants rate the technique. The mean results for these ratings can be seen in Figure 5.11. These results match what we had expected, with Highlights being the highest, followed by Compass, then Hint and None. What we did not expect, however, is how close Compass would be to Highlights.

Our statistical analysis on these ratings show that certain pair differences were not significant. The pairs were Hint/None for both games, and Highlights/Compass for Third-Person. During the experiment, we observed how participants filled out the rating table. More often than not, participants would rank Highlights and Compass first as their highest ratings (best and second best), and then seemingly carelessly place Hint and None as the bottom two. As for the Highlights/Compass pair, we think it is due to the fact players preferred them equally and therefore their ratings are not significant. This is also reflected in the preference pie charts (see Figure 5.12) and how participants spoke about the two techniques, both in person and in their questionnaire comments.
6.2 - “Highlight” Implementation Differences

Out of the three assistance techniques implemented for evaluation, Highlights, was the only one that was visually distinct between the two game types. Our aim was to have a strong implementation of Highlights in both games.

For the Top-Down game, a simple and straightforward outline around the target, as seen in Figure 6.1 (Left), was sufficient to achieve a strong implementation, while in Third-Person a similar implementation was far weaker, as the environmental obstacles quite often obstructed the players’ view of the target and its highlight effect as – not – seen in Figure 6.1 (Middle).

By altering the visual design of the highlight in the Third-Person game to a flare-like particle effect as seen in Figure 6.1 (Right), players would then be able to further benefit from the technique as it overcomes the problem of obstruction since the flare effect is taller than most of the environmental objects in the scene.

![Figure 6.1 (Left) Top-Down Highlights. (Middle) Third-Person Highlights original implementation, similar to that of Top-Down to the left. (Right) Third-Person new Flare-Highlights effect.](image)
6.3 - Two-Way Performance Adjustment: “Negative Assistance”

Player performance may be adjusted positively – *assisted* – so that certain attributes such as the players’ scores, speed, and/or accuracy are improved. Similarly, player performance could be adjusted negatively – *hindered* – to diminish those same attributes.

Most of our discussions up to this point have focused on providing assistance and *improving* player performance. However, it is important to remember that many of the techniques outlined for the core tasks may be used to create a range of difficulties.

Noise Reduction is meant to help with Signal Detection and Visual Search tasks, but applying the technique in reverse – i.e., *adding* more noise – can make the task more difficult. The same can be said for techniques like Target Gravity or Steering Adjustment.

However, several techniques cannot be applied negatively, and in such situations designers may implement the technique when the task needs to be easy (Easy mode, early game levels, etc.), and reduce its strength to adjust difficulty as required. In such cases, the lowest possible strength would be no implementation at all.

An example of this is Advance Warning for Reaction Time, where a strong implementation would be a warning several seconds before player reaction is required, and the weakest would be no warning at all.
6.4 – How do the study results inform game designers about incorporating assistance into their games?

Core task identification is a vital step in deciding which assistance techniques to pursue. In our evaluation, the games were comprised of a handful of core tasks, mainly Visual Search and Pointing for both games, and Steering for the Third-Person game.

Once the core tasks are identified, the appropriate assistance can be chosen to adjust player performance. If the desired effect or difficulty is not achieved, designers may then consider providing assistance to the other tasks in the game. For example, in our evaluation, players could have benefitted from a Pointing assistance technique to improve their scores even further.

Game designers need to test out their techniques carefully. Sometimes the evaluation results followed conventional wisdom, but sometimes we found unexpected results. For example, Compass performed nearly as well as Highlights, especially in Third-Person.

Player preference is not always about performance. Assistance techniques can make certain tasks too easy, as we saw with Highlights. Even though it was effective and increased the players’ scores considerably, players often preferred the Compass, which did not always have the same effect as Highlights.

Designers should keep this in mind when deciding when and where to implement certain techniques and evaluate the level of difficulty required clearly and choose an appropriate technique accordingly.
7 – Conclusion and Future Work

7.1 - Conclusion

In this thesis, we explored 10 different Core Tasks of gaming, discussing each of the possible assistance techniques for all of them. A visual summary of our organization can be seen in Figure 4.1. Several of those techniques have been explored academically before, while others are still to be explored and evaluated.

By looking at video game assistance at a fundamental level like we do here, through the lens of the Core Tasks, we assist in the portability and understanding of these techniques across games, regardless of genre or platform.

One of the main goals of this thesis was to create a comprehensive starting point for designers and game developers considering assistance for their games. We have successfully collected and presented 35 assistance techniques from various sources and defined a clearer language for assistance discussions. We defined the core tasks in terms of video games, and outlined the process for analyzing games to identify their core tasks.

We also conducted a study on the effectiveness of several techniques pertaining to Visual Search: Highlights, Target Details, and Compass. The results of the experiment show that the techniques do in fact have an effect on performance. We also demonstrated the effectiveness of our generalized model by implementing those three techniques into three different types of games: Top-Down (puzzle-like games), Third-Person (adventure and RPG-like games), and a Sniper simulation.
Assistance techniques are widely used in video games to help players have the most engaging and/or enjoyable experience possible. Our hope is that this work provides game designers with a valuable resource for applying assistance techniques, identifying assistance techniques and discovering new assistance techniques that can be applied in games. Ultimately, we hope our work contributes to the advancement of game design and can help improve entertainment experiences.

7.2 - Future Work

In this section, we describe some possible directions for future work.

7.2.1 - Refine List of Core Tasks

We believe that the list of core tasks presented in this thesis is comprehensive and concise in terms of the most fundamental actions that players may perform in games. Details on how the list was generated and evaluated can be found near the beginning of the thesis (see 3.1 - Core Tasks).

At some point, however, the list may become outdated as video games move on to more advanced and innovative platforms that could manifest a whole new way of interacting with games. For example, advancements in Brain Computer Interfaces (BCI) could one day see such technology available to game developers and the entertainment market. We would then need to discuss assistance in terms of players’ thoughts as input.

Because the core tasks are based on the most fundamental actions players may perform in games, it is possible that other core tasks exist under different levels of granularity of
these same fundamental actions. It is possible that the differences within one core task become significant enough to require splitting it up into unique categories.

For example, Body Controls is currently a comprehensive category that includes Muscle Activation, Ambidexterity and Movement. These subcategories may become more distinct as games begin to take advantage of body input, especially with the advancements made in Virtual and Augmented Reality technologies [64].

The current list of core tasks still has value, even if it may be incomplete for all future possibilities, as it can help us identify design options and provides us with a language to discuss different possibilities.

**7.2.2 - Expand List of Assistance Techniques**

We believe that our work has provided us with a wide range of known assistance techniques for each of the core tasks we identified.

It is possible, however, that some techniques have gone unnoticed. We already did an extensive analysis of many video games to finalize our list of Core Tasks, compile relevant assistance techniques, and show examples of commercial games incorporating said techniques (See 3.1 - Core Tasks and 3.2.8 - Methodology). However, there is a plethora of games available, few of which have assistance techniques implemented. Even fewer make it explicit that assistance is available.

For example, Overwatch is packed with assistance techniques by default, so many in fact that it can be used as an example for nearly all the techniques discussed in this paper
(perhaps not Body Controls). However, the only assistance that is explicitly mentioned is their aim assist, which is technically a Pointing assistance technique.

Other work may focus on developing brand new techniques, or perhaps variations of existing techniques that are sufficiently unique to be considered novel techniques. For example, this is likely to occur with Pointing tasks, as most of the techniques listed there are technically a combination of Fitts’s Law adjustments.

It has been shown that pointing tasks in user interfaces can be improved or made easier by adjusting the target width (size) or the distance from the pointer to the target, or both [47]. The different techniques discussed for Pointing can be imagined as labels for the combinations of adjustments on size and distance that make it easier for us to discuss them. The techniques that we have listed are ones that we have observed in commercial games or ones that were developed for an alternative purpose, such as pointing on a desktop, that we concluded could fit within a game environment.

However, it may be possible for other techniques to be discovered as well. The purpose of this work was not to develop new assistance techniques, but rather to identify existing techniques and to better describe how they work based upon the notion of Core Tasks.
7.2.3 - Further exploration and analysis of assistance techniques

7.2.3.1 - Explore previously unexplored techniques

One of the issues with previous work in assistance is that not all the techniques have been explored and evaluated scientifically. A few have never been explored in such a way. Therefore, future research could take a deeper look at the effectiveness of these techniques and evaluate them to further our understanding of their potential, differences, and drawbacks.

7.2.3.2 - Explore techniques in various situations

Another one of the issues with previous work is the narrow scope of evaluation, usually within one type of game and platform. Our work demonstrates how assessing techniques in different scenarios is important, because different techniques can and do perform differently in different types of games. Future work may attempt to explore these same techniques in other types of games, like how we evaluated the Visual Search techniques in top-down and third-person games. Perhaps another direction would be to explore those techniques on different platforms, such as Pointing techniques in a virtual reality environment rather than a console or PC (Mouse and Keyboard) setting.

7.2.3.3 - Evaluate combinations of techniques for a Core Task

During our evaluation of the Visual Search techniques, we tested the effects of each of the techniques individually. Future work may explore the effect of having two or more of the techniques enabled simultaneously. For example, we would expect Compass and Highlights to be more effective in the Third-Person version of the game than either of
them individually, and it would be interesting to see if that is indeed the case. It would also be interesting to find out whether certain combinations of techniques have a detrimental effect on player performance.

7.2.3.4 - Evaluate combinations of techniques across different Core Tasks

Similar to the previous point, future work should look into combining techniques from different core tasks in games that include two or more core tasks. In our evaluation, we often observed that when players identified the target, they would rush to click on it and sometimes miss it and click right next to it. This wastes time and may even result in the player clicking an incorrect target if it was nearby. We expect that a Pointing technique such as Target Gravity would mitigate such a problem.
References


http://wtbgold.blogspot.com/2014/06/node-changes-in-warlords-of-draenor.html#axzz5OekD6IO5


Referenced Media


Ludography


https://en.wikipedia.org/wiki/Street_Fighter

https://en.wikipedia.org/wiki/Tekken

https://en.wikipedia.org/wiki/Metal_Gear

https://en.wikipedia.org/wiki/Sly_Cooper


[L13] Rockstar Games, 1997, *Grand Theft Auto*, PlayStation, Rockstar Games, USA.  
https://en.wikipedia.org/wiki/Grand_Theft_Auto

https://en.wikipedia.org/wiki/Monster_Hunter

[L15] Turn 10 Studios, 2005, *Forza Motorsport*, Xbox, Microsoft Game Studios, USA.  
https://en.wikipedia.org/wiki/Forza_(series)

https://en.wikipedia.org/wiki/Monster_Hunter:_World

https://en.wikipedia.org/wiki/Dead_by_Daylight


https://en.wikipedia.org/wiki/Guitar_Hero

https://store.steampowered.com/app/643270/OLDTV/


Appendix A - Referenced Games

This is a list of some of the games referenced in this document. Each game is briefly described and relevant parts of the game are highlighted to provide additional context for the discussions above.

**Monster Hunter: World** is an action role-playing game where players use a variety of weapons, items, abilities, and tactics to hunt down large monsters, either solo or in multiplayer groups.

One of the weapons available to players is the *Great Sword*, which can be used to perform a *Charged Attack* [M4], which grows in power with every tick (or stage) of the charge to a maximum of three ticks. The power of the attack is maximized if the player lets go at the same time as a tick.

One technique that players can use to weaken their prey is to *mount* it, deal a small amount of damage to it and potentially knock it down. While the monster is mounted, it will attempted to throw off the player or smash them into the environment. The player must pay attention to the creature’s actions and react accordingly by either *bracing*, *moving*, or *attacking*.

**Dead by Daylight** is an asymmetric survival horror online multiplayer game where 5 players take on the roles of one *Killer* and four *Survivors*. The survivors’ goal is to avoid detection, complete objectives, and escape alive. The killer tries to find the survivors, capture them, and kill them.
Dead by Daylight includes a system of Perks, which are toggle-able power-ups for both survivors and killers. These perks are varied with unique effects, such as Spies from the Shadows described in the example for the Companion Signals technique.

**Overwatch** is a – mostly competitive – first-person shooter game where players take on the role of one of many characters, each with a unique weapon.

**World of Warcraft** is a massively multiplayer online role-playing game in which some of the most popular activities are Dungeons and Raids. A raid [5] is a group activity where several players engage in virtual combat against a power enemy usually referred to as the boss.

Dungeons and raids vary in difficulty; however, the most significant ones are the ones attempted by the maximum-level players during the ‘end-game’ as there are no short cuts and progress is often slow. It is not uncommon to see groups of players spend weeks simply trying to defeat one boss [6]. Therefore, players at this level are expected to play with a certain degree of skill.

World of Warcraft also allows third party Add-ons [7], which are software tools installed independently by players that alter the game experience in some way, such as UI improvements, etc., without changing the functionality of the game.
## Appendix B – Tables and Questionnaires

Table B-1 Non-exhaustive list of video game genres with examples of popular games for each one.

Core tasks are listed for each one based on player tasks/actions found in their respective examples.

<table>
<thead>
<tr>
<th>Genre</th>
<th>Subgenre</th>
<th>Example</th>
<th>Core Task(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Platformer</td>
<td>Donkey Kong</td>
<td>Reaction Time, Steering</td>
</tr>
<tr>
<td>Action</td>
<td>Platformer</td>
<td>Super Mario</td>
<td>Aiming, Reaction Time, Steering</td>
</tr>
<tr>
<td>Action</td>
<td>Shooter</td>
<td>Call of Duty, Overwatch</td>
<td>Aiming, Reaction Time, Steering, Pointing, Signal Detection, Signal Discrimination, Visual Search, Spatial Memory</td>
</tr>
<tr>
<td>Action</td>
<td>Fighting</td>
<td>Street Fighter, Tekken</td>
<td>Signal Detection, Reaction Time, Steering</td>
</tr>
<tr>
<td>Action</td>
<td>Stealth</td>
<td>Metal Gear, Sly Cooper</td>
<td>Signal Detection, Signal Discrimination, Reaction Time, Steering, Aiming, Pointing, Spatial Memory</td>
</tr>
<tr>
<td>Action</td>
<td>Survival</td>
<td>Resident Evil</td>
<td>Signal Detection, Signal Discrimination, Reaction Time, Steering,</td>
</tr>
<tr>
<td>Action</td>
<td>Adventure</td>
<td>-</td>
<td>Spatial Memory, Aiming, Pointing</td>
</tr>
<tr>
<td>Adventure</td>
<td>-</td>
<td>Myst</td>
<td>Steering, Pointing, Spatial Memory, Visual Search, Signal Detection</td>
</tr>
<tr>
<td>Roleplaying</td>
<td>MMORPG</td>
<td>World of Warcraft</td>
<td>Signal Detection, Signal Discrimination, Reaction Time, Steering, Pointing, Aiming, Visual Search, Pattern Matching, Short Term Memory, Spatial Memory</td>
</tr>
<tr>
<td>Roleplaying</td>
<td>Sandbox RPG</td>
<td>Grand Theft Auto</td>
<td>Signal Detection, Signal Discrimination, Reaction Time, Steering, Pointing, Aiming, Visual Search, Pattern Matching, Short Term Memory, Spatial Memory</td>
</tr>
<tr>
<td>Simulation</td>
<td>-</td>
<td>Sims</td>
<td>Signal Detection, Signal Discrimination, Pointing, Visual Search, Pattern Matching, Spatial Memory</td>
</tr>
<tr>
<td>Strategy</td>
<td>Real-Time Strategy (RTS)</td>
<td>Starcraft II</td>
<td>Signal Detection, Signal Discrimination, Reaction Time, Pointing, Visual Search, Short Term Memory, Spatial Memory</td>
</tr>
<tr>
<td>Strategy</td>
<td>MOBA</td>
<td>League of Legends</td>
<td>Signal Detection, Signal Discrimination, Reaction Time, Pointing, Aiming, Visual Search, Short Term Memory, Spatial Memory</td>
</tr>
<tr>
<td>Sports</td>
<td>Racing</td>
<td>Forza Motorsports</td>
<td>Signal Detection, Reaction Time, Steering, Aiming, Visual Search, Spatial Memory</td>
</tr>
<tr>
<td>Sports</td>
<td>-</td>
<td>Rocket League</td>
<td>Signal Detection, Signal Discrimination, Reaction Time, Steering, Aiming, Visual Search, Spatial Memory</td>
</tr>
</tbody>
</table>
Table B-2 A short snippet of Overall test data collection.
These particular entries are not from the experiment results.

<table>
<thead>
<tr>
<th>I.D.</th>
<th>Game Type</th>
<th>Assistance Type</th>
<th>Score</th>
<th>Targets Found</th>
<th>Mean Time Per Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>ThirdPerson</td>
<td>Compass</td>
<td>18</td>
<td>10</td>
<td>5.47</td>
</tr>
<tr>
<td>13</td>
<td>ThirdPerson</td>
<td>Highlight</td>
<td>2</td>
<td>2</td>
<td>25.98</td>
</tr>
<tr>
<td>13</td>
<td>ThirdPerson</td>
<td>Compass</td>
<td>17</td>
<td>10</td>
<td>5.51</td>
</tr>
<tr>
<td>33</td>
<td>TopDown</td>
<td>Highlight</td>
<td>74</td>
<td>38</td>
<td>1.63</td>
</tr>
<tr>
<td>33</td>
<td>ThirdPerson</td>
<td>Highlight</td>
<td>10</td>
<td>6</td>
<td>9.61</td>
</tr>
</tbody>
</table>

Table B-3 A short snippet of Per-Click test data collection.
These particular entries are not from the experiment results.

<table>
<thead>
<tr>
<th>I.D.</th>
<th>Game Type</th>
<th>Assistance Type</th>
<th>Target Shape</th>
<th>Target Color</th>
<th># of Clicks</th>
<th>Start Time</th>
<th>End Time</th>
<th>Previous Target</th>
<th>Next Target</th>
<th>Dist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Top Down</td>
<td>Highlight</td>
<td>Pentagon</td>
<td>Red</td>
<td>1</td>
<td>16.06</td>
<td>17</td>
<td>(-12.0, 1.1, -30.9)</td>
<td>(-18.5, 1.2, -2.1)</td>
<td>29.5019</td>
</tr>
<tr>
<td>15</td>
<td>Top Down</td>
<td>Highlight</td>
<td>Heart</td>
<td>Pink</td>
<td>1</td>
<td>17</td>
<td>17.96</td>
<td>-18.5, 1.2, -2.1</td>
<td>-47.1, 1.1, 23.6</td>
<td>38.5477</td>
</tr>
<tr>
<td>15</td>
<td>Top Down</td>
<td>Highlight</td>
<td>Cube</td>
<td>Pink</td>
<td>1</td>
<td>17.96</td>
<td>18.88</td>
<td>-47.1, 1.1, 23.6</td>
<td>-4.7, 1.4, 3.8</td>
<td>46.8439</td>
</tr>
<tr>
<td>15</td>
<td>Third Person</td>
<td>Compass</td>
<td>Cube</td>
<td>White</td>
<td>1</td>
<td>35.12</td>
<td>37.46</td>
<td>20.1, 0.9, 3.2</td>
<td>0.7, 1.4, 11.0</td>
<td>22.2272</td>
</tr>
<tr>
<td>15</td>
<td>Third Person</td>
<td>Compass</td>
<td>Ball</td>
<td>Red</td>
<td>3</td>
<td>37.46</td>
<td>42.88</td>
<td>-0.7, 1.4, 11.0</td>
<td>-0.4, 2.2, 10.3</td>
<td>1.14366</td>
</tr>
<tr>
<td>Question</td>
<td>Scale</td>
<td>Response Options</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Playing the game was mentally demanding.</td>
<td>1-7</td>
<td>Strongly disagree 1 2 3 4 5 6 7 Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Playing the game was physically demanding.</td>
<td>1-7</td>
<td>Strongly disagree 1 2 3 4 5 6 7 Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt rushed while playing the game.</td>
<td>1-7</td>
<td>Strongly disagree 1 2 3 4 5 6 7 Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I performed well during my recent attempt.</td>
<td>1-7</td>
<td>Strongly disagree 1 2 3 4 5 6 7 Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I had to try hard to achieve this level of performance.</td>
<td>1-7</td>
<td>Strongly disagree 1 2 3 4 5 6 7 Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I was discouraged, irritated, annoyed, or stressed while playing.</td>
<td>1-7</td>
<td>Strongly disagree 1 2 3 4 5 6 7 Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I would like to see similar assistance techniques implemented in video games.</td>
<td>1-7</td>
<td>Strongly disagree 1 2 3 4 5 6 7 Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I have seen a similar technique in games before.</td>
<td>1-7</td>
<td>Strongly disagree 1 2 3 4 5 6 7 Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I had fun playing this game.</td>
<td>1-7</td>
<td>Strongly disagree 1 2 3 4 5 6 7 Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I am pretty skilled at this game.</td>
<td>1-7</td>
<td>Strongly disagree 1 2 3 4 5 6 7 Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure B-1 Questionnaire: Pre-Experiment**  
**Figure B-2 Questionnaire: Post Technique**
Figure B-3 Questionnaire: Post Game

**The game was easier with assistance techniques.**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In what type of game, or game scenario, would you imagine such techniques implemented? *

Your answer

I preferred playing with an assistance technique enabled.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Would you like to see such techniques implemented in games wherever appropriate? *

Your answer

I feel that such techniques can be useful in similar games.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly disagree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strongly agree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If assistance is being used in a game you are playing, would you like to be told about it? *

Your answer

Rank the techniques in terms of their ability to help your performance. (1 = Most helpful) *

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highlight</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hint</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compass</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Would you like assistance to be applied automatically? Or would you prefer an option to enable/disable it yourself? *

Your answer

Which technique did you feel best fit with this type of game? *

- Highlights
- Hint
- Compass

Figure B-4 Questionnaire: End of Experiment

Comments

Your answer
Appendix C - Curriculum Vitae

Candidate's full name: Jawad Jandali Refai

Universities attended:

University of New Brunswick – New Brunswick, Canada
September 2016 to October 2018
Master of Computer Science (MCS)

University of Wollongong – Dubai, United Arab Emirates
September 2012 to June 2016
Bachelor of Science in Computer Science (B.Sc.) with Distinction

Publications: None

Conference Presentations: None