



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Situated Visualization in Motion for Video games

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE ENGINEERING - INGEGNERIA INFORMATICA

Author: **Federica Bucchieri**

Student ID: 968132

Advisor: Prof. Franca Garzotto

External advisors: Petra Isenberg, Lijie Yao

Academic Year: 2021-22

Abstract

Video games produce rich dynamic datasets during gameplay that are often visualized to help players succeed in the game. Typically, these visualizations are moving either because they are attached to moving game elements or due to camera changes. The goal of my thesis is to understand to what extent this motion and contextual game factors impact how players can read these visualizations. First, I contribute a systematic review of situated visualizations in motion in the context of video games, categorizing 160 visualizations collected by analyzing 50 video games into multiple dimensions, including the visual representation used, the data referent, the color appearance and the embedding locations. Second, I contribute an analysis of those visualizations for different types of data, with a focus on quantitative and categorical data representations. Third, I contribute the implementation of a video game called RobotLife, which is a FPS video game for PC that contains embedded visualizations with the game character. Finally, I conducted a extensive pilot study about the impact of different embedding locations on three visualizations' readability from different aspects including visual design and motion factors. Results showed that visualizations embedded around the game element were more visible while in motion, with respect to visualizations overlapping the game element or integrated in the game element's design. Furthermore, integrated visualizations were more immersive and therefore perceived as more beautiful. Overlapping visualizations represented a good trade-off between readability and aesthetic, particularly if they fit the game element design. To conclude, motion factors can reduce the visualizations' readability, but it also depends on the participants' gaming expertise.

Keywords: data visualization, visualization in motion, video games visualization, situated visualization

Abstract in lingua italiana

I videogiochi producono ricchi set di dati durante le sessioni di gioco. Tali dati sono spesso visualizzati per aiutare i giocatori a vincere. Tipicamente, queste visualizationi sono in movimento sullo schermo sia perchè attaccate ad elementi di gioco in movimento, sia a causa di cambiamenti nell'angolazione o rotazione della camera di gioco. Lo scopo della mia tesi è comprendere fino a che punto fattori di movimento e di contesto di gioco impattano la leggibilità di tali visualizzazioni. Innanzitutto, ho condotto una revisione sistematica di visualizationi situate in movimento, nel contesto dei videogiochi. Analizzando 50 videogiochi, ho raccolto 160 visualizzazioni catalogandole secondo dimensioni tra cui il tipo di rappresentazione grafica utilizzata, il referente del dato, il colore utilizzato e la posizione di incorporamento rispetto al referente del dato. A seguire, ho condotto un'analisi di tali visualizzazioni a seconda del tipo di dato che rappresentavano, con maggiore attenzione sulle rappresentazioni di dati quantitativi e categoriali. Ancora, ho implementato un videogioco chiamato RobotLife, un gioco sparatutto in prima persona sviluppato per PC, che contiene visualizationi di dati incorporate nei personaggi del gioco. Infine, ho condotto uno studio riguardo l'impatto di diverse posizioni di incorporamento dei dati, analizzando la leggibilità di tre diverse visualizzazioni a seconda di vari aspetti tra cui il design grafico e i fattori di movimento. I risultati mostrano che le visualizzazioni incorporate intorno agli elementi di gioco sono più visibili in movimento rispetto a visualizzazioni che si sovrappongono agli elementi di gioco, o che sono integrati nel loro design. In aggiunta, le visualizzazioni di dati integrate con gli elementi di gioco risultano più immersive e per questo più piacevoli ed esteticamente belle. Le visualizzazioni di dati che si sovrappongono agli elementi di gioco invece rappresentano un buono compromesso tra leggibilità ed estetica, soprattutto se il loro design si amalgama bene con il design dell'elemento di gioco al quale sono incorporate. Per concludere, i fattori legati al movimento delle visualizzazioni sullo schermo posso ridurre la leggibilità delle visualizzazioni di dati. Questo però dipende anche dall'esperienza dei giocatori stessi.

Parole chiave: Visualizzazione di dati, visualizzazioni in movimento, visualizzazioni nei videogiochi, visualizzazioni situate.

Contents

Abstract	i
Abstract in lingua italiana	iii
Contents	v
1 Introduction	1
1.1 Motivation	2
1.2 Research Context	2
1.3 Research Goals	3
1.4 Contributions	4
2 Background	5
2.1 Situated Visualization	5
2.2 Visualization in Motion	6
2.3 Video Games Visualization	8
3 A Systematic Review of Visualizations in Motion in Video Games	11
3.1 Video Games Selection	11
3.2 Categorization of Current Video Game Visualizations	12
3.3 Relationships between Different Dimensions	17
4 Analysis of different types of data in video games	21
4.1 How different Types of Data are represented in Video Games	22
4.2 Character's Health	23
4.2.1 Analysis of character's health representation	24
4.3 Game Element Types	25
4.3.1 Analysis of game element types representation	26
4.4 Summary	26

5 RobotLife: A Video game for the Evaluation of Visualization in Motion	29
5.1 First Person Shooter games	29
5.2 Implementation Environment	30
5.3 RobotLife Storyline	31
5.4 RobotLife Characters: The Hoverbots	32
5.5 Game Specifics	32
5.6 Tutorial	34
6 A Study Design for Visualization in Motion in Games	35
6.1 Contextual Factor to Evaluate	35
6.2 Choice of Data to Embedded in the Characters	36
6.3 Choice of Visualizations to Evaluate	37
6.4 Choice of Percentages	38
6.5 Training Session	39
6.6 Gameplay Flow	39
7 RobotLife Implementation	43
7.1 Game World Implementation	43
7.2 Game Mechanics Implementation	45
7.2.1 ObjectiveKillEvilRobots.cs	45
7.2.2 ObjectiveSaveGoodRobots.cs	45
7.2.3 EnemyController.cs	46
7.2.4 EnemyMobile.cs	46
7.3 Visualizations Implementation	47
7.4 Tutorial Implementation	48
7.5 Second Level: Game element types visualizations	50
8 Pilot Study	55
8.1 Pre-Pilot	55
8.2 Participants	56
8.3 Experimental Software and Apparatus	57
8.4 Study Procedure	57
8.5 Analysis Approach	58
9 Results	61
9.1 Game performances	61
9.2 In-game questionnaire results	62
9.2.1 Readability	62

9.2.2	Aesthetic	64
9.2.3	Summary	65
9.3	Post-questionnaire results	65
9.3.1	Summary	66
9.4	Interview results	68
9.4.1	Readability	68
9.4.2	Aesthetic	69
9.4.3	Embedding Locations	70
9.4.4	Hover bot motion	71
9.4.5	Self-motion	71
9.4.6	Strategies	71
9.5	Discussion	72
10	Considerations and Limitations	73
10.1	Design Considerations	73
10.2	Limitations	75
11	Conclusions	77
11.1	Future Work	78
Bibliography		79
A	Appendix A	85
A.1	Systematic review: Selected Video Games	86
A.2	Systematic review: Data collected	87
A.3	Systematic review: Relationships between different dimensions	94
A.4	Visualization design sketches	96
A.5	Game mechanics implementation scripts	99
A.6	Visualization Implementation: Colored texture Shader	117
A.7	Pre-Questionnaire	119
A.8	In-game Questionnaire	121
A.9	Post-Questionnaire	122
A.10	Pilot Study results	124
List of Figures		129

List of Tables	133
Acknowledgements	135

1 | Introduction

Video games produce rich dynamic datasets during gameplay that are often visualized to help players succeed in a game and make important decisions about how to act in the game. Examples include health bars, navigation aids, ammunition count, or team affiliation (see Figure 1.1). As such, visualizations play an important role in how effective a player can be but they also pose a number of interesting design challenges. Visualizations need to be read at a glance while the player is focused on a primary task such as fulfilling a game mission. Also, they often need to be small, match the aesthetics of the game, or be closely embedded next to game elements [6]. Frequently, those data visualizations are in motion on the screen due to camera changes or because the visualizations are attached to moving game elements such as game characters. This thesis aims to explore how contextual factors in video games affect the readability of visualizations in motion. In this section I present the motivation for my research, give an overview of the research context and goal and provide more insight into the inherent problems in the design of visualization in motion for video games.



(a) Navigation aids and objective location in *Watch Dogs: Legion*.



(b) Ammunition count embedded in the weapon display in *Halo Infinite*.



(c) Objectives' team affiliation and soldiers' death positions in *Call of Duty: Black Ops II*.

Figure 1.1: Examples of situated visualizations in motion in the context of video games.

1.1. Motivation

Movement is a crucial aspect of many types of video games and as such, many people are used to follow moving game elements on the screen. The presence of data related to such game elements is sometimes crucial for the player to act in the game. Designing visualization subject to motion factors is challenging because, compared to standard desktop visualizations many contextual factors are introduced such as speed, direction and trajectory of motion, size of the visualizations and background changes. For example, we can consider a circular bar chart displayed below the feet of a character that presents the character's stamina. The stamina value can be harder to read if the character jumps, continuously changes direction and varies its speed. Moreover, if the bar is colored in white and the character moves from a green countryside scenario to a snowy mountain scenario, the likelihood of the visualization to be read correctly decreases as a consequence of the low color contrast between the bar and background color [56]. Understanding how motion factors influence the readability of visualization in motion in the context of video games can help game designers improve data visualization and subsequently player performance and enjoyment in their games. Previous research on visualization techniques in the context of video games analyzed how data is presented to players. This thesis extends this past work in that it concentrates on the motion of visualizations related to contextual factors of the game itself which have not yet been explored.

1.2. Research Context

To frame my thesis in the current scientific field, I now outline the research areas my work belongs to and narrow them to define the scope of this thesis. On a very high level, my work is in the Human Computer Interaction (HCI) domain. Specifically, since this thesis tackles the problem of presenting data to video game users, it lies in the information visualization (InfoVis) field. Information visualization can be defined as the study of visual representations to explore abstract data with the ultimate goal of helping users to understand the data better, extract patterns and relations between them [7]. InfoVis is an interdisciplinary field of study and comprises the three main research areas that play a role in my thesis: Situated Visualization, Visualization in Motion and Video Game Visualization. Situated visualizations are those visualizations that display data in proximity to data referents [63][53]. As an example, icons showing the team affiliation displayed above the head of a character are considered situated visualizations as they are placed in proximity with the data referent (the character). A data referent is defined as the entity the data refers to. This thesis also lies in the context of video game visualization

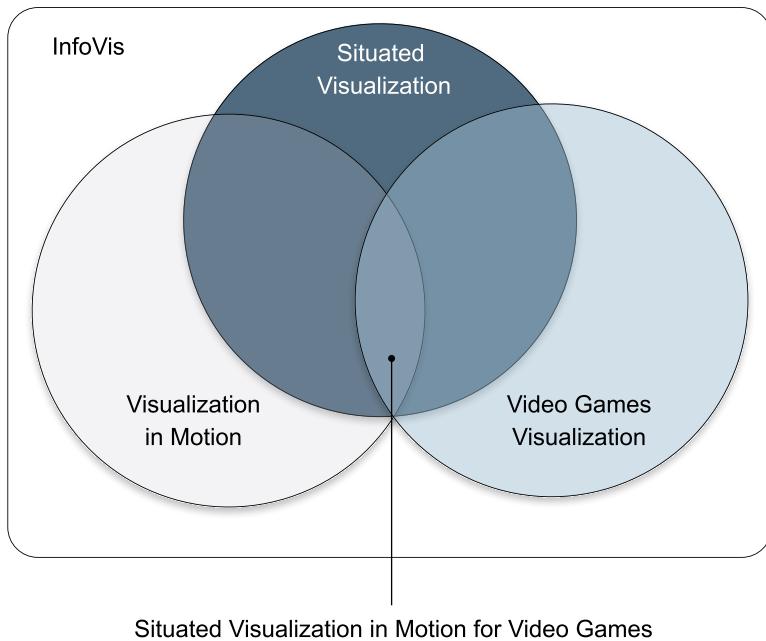


Figure 1.2: The research context of this thesis lies in the intersection between the Situated Visualizations, Visualization in Motion and Video Games Visualization domains.

since I focus on video game environments and in-game visualizations. Lastly, my research builds on the area of Visualization in Motion. This research area is particularly new. Yao et al. [66][64] proposed the first design space of Visualization in Motion, opening new research possibilities in the field. They define visualizations in motion as "visual data representations used in contexts that exhibit relative motion between a viewer and an entire visualization". Framing the research context in the intersection between the three aforementioned areas of InfoVis, my work has three main constraints. First, I only consider visualization placed in close proximity with the data referent. Second, I explore visualization in the specific context of video games. Last, I only analyze visualizations that present relative motion with respect to a static viewer, the player.

1.3. Research Goals

In the context outlined above, my first goal is to acquire a general understanding of the existing practices of situated visualization in motion for video games. Moreover, I pursue the following research goals.

- **Understanding embedded visualization choices and their effect on video game players**

My goal is deriving a study design about the design choices in embedding visualiza-

tions with game elements in motion by defining a set of contextual factors affecting the visualizations readability and analyzing them. The contextual factors will focus on motion and situated visualization characteristics.

- **Implementing an evaluation program**

The final goal of my thesis is to implement a video game to evaluate different visualizations with respect to the aforementioned research goals. This game will contain game elements that can act as data referents, different types of visualizations and tutorials and game mechanics that allow to compare different designs.

1.4. Contributions

To achieve the presented research goals, my contributions are the following.

- **A systematic review of situated visualizations in motion in the context of video games**

Through a survey of 160 visualizations in motion and their embeddings in the game world, I provide an analysis and categorization of these visualizations. The categorization comprises several dimensions related to situated visualization and motion factors.

- **An analysis of situated visualizations in motion in video games for different types of data**

In addition to the systematic review, I contribute an analysis on how different types of data are represented in games. Specifically, I focused on quantitative and categorical data by analyzing two prominent examples of the information displayed in video games: character's health and the type of specific game elements.

- **A video game for the evaluation of situated visualization in motion for video games**

I designed and implemented a First Person Shooter (FPS) video game to conduct an empirical evaluation of different visualizations in motion. The video game is implemented in Unity and deployed for PC. I developed the game design, level design, gameplay design, visualization design, implementation and game testing.

- **A study on visualization in motion in video games**

Relying on the video game I implemented, I contribute a study on three different visualizations in motion embedded with the game characters. The study evaluates visualizations with different embedding locations and how they affected players' experience.

2 | Background

Nowadays, visual representations such as tables, charts and maps are widely used in everyday life. The domain of data visualization studies the visual representation of data in order to help users recognize patterns and information in data [30]. Creating visualization comes with various design challenges such as choosing the correct visual representation, the color, or size and moreover designing the right visualization having in mind the context and the audience it is intended for [21]. This thesis builds on three different research areas within data visualization: Situated Visualization, Visualization in Motion and Video Games Visualizations. This chapter presents previous research in those three domains.

2.1. Situated Visualization

Situated visualization are those data representations that are located in close proximity to relevant referents such as objects, locations, and people in the physical world. Examples of situated visualizations could be representations displayed in augmented reality directly overlapping the object of interest or displaying data about a shop on its front door. New technologies such as mixed reality, wireless displays and sensors make it easier to display data in context. White and Feiner [58, 59] described situated visualizations as "*visualization that are relevant to the location or physical context in which they are displayed*". They studied different presentation and interaction techniques, investigating a theoretical framework and best practices for situated visualizations. Of particular interest in their work was the visualization's environment. Their framework's pillars comprised: spatial and temporal relevance, locus of presentation and interaction techniques. Those pillars are taken into consideration in my designs. The dimension of embedding locations is tightly related to White and Feiner's locus of presentation and spatial relevance.

Thomas et al. [16, 53] introduced Situated Analytics (SA) as the use of situated visualizations aimed at supporting analytic reasoning. They presented design considerations for situated analytics applications based on a set of case studies analyzed. Thomas et al. explored the concept of spatially situated visualizations, analyzing the differences between physical and perceptual distance between representation and data referent. Moreover,

they discussed embedded visualizations defined as "*situated visualization that are deeply integrated with their physical environment*".

Elaborating on the concept of embedded visualization, Willett et al. [63] formalized the difference between situated and embedded visualization, analyzing the relationship among data and data referents. In situated data representation we find a relationship of spatial proximity. Embedded representation take this proximity to the extreme by overlapping data to the referent itself. Examples of embedding visualizations are overlays, projections and see-through videos. In my design I made use of both situated and embedded visualizations with the game characters, considering the proposed definitions.

Specific instances of situated visualizations have also been discussed in the literature, for example in the domain of urban visualization [8, 43]. Vande Moere and Hill [54] investigated the concept of urban data that has a contextual relationship with the surrounding environment, within the public space of the city, with the final aim of communicating actionable information to local inhabitants. Visualizing information in everyday urban context is not a trivial task. In their work, Vande Moere and Hill presented the best practises for urban visualization by analyzing some examples such as the U.S. Debt Clock in New York [61] and the Cykelbarmeter in Copenhagen [60]. Vande Moere and Hill considered the embedding in the real world as a fundamental characteristic of urban visualization.

My thesis focuses specifically on visualizations that are displayed in proximity with the data referents or embedded with it. For example when information about a game character is displayed directly next to the character itself. Here the concept of physical proximity is considered to persist as close distance of elements in the virtual world.

2.2. Visualization in Motion

As defined by Yao et al. [64, 66] visualization in motion studies visualizations that present relative motion between the viewer and the visualization. Yao et al. identified three possible scenarios where the readability of the visualization is influenced by motion factors: moving visualizations & stationary viewers, stationary visualizations & moving viewers,

	Viewer stationary	Viewer moving
Vis stationary		x
Vis moving	x	x

Table 2.1: Different scenarios of visualization in motion identified by Yao et al. [66]. The work of this thesis falls into the scenario highlighted in blue: viewer stationary and visualization moving.

and moving visualizations & moving viewers (Table 2.1). When playing video games this motion relationship involves a stationary player sitting in front of a monitor, seeing visualizations moving on screen. In order to read visualizations accurately in this scenario, the players are required to move their eye and/or head.

As part of a research agenda, Yao et al discussed characteristics of motion that required further research including characteristics to describe physical motion. Some of those characteristics are speed, trajectory, acceleration and direction of motion. For example, speed and trajectory are important in racing and sports games, as well as acceleration and direction of motion can be relevant in First Person Shooter (FPS) and Third Person Shooter (TPS) games. They discussed the spatial relationship between the viewer and the visualization such as the viewing distance that in the context of video game can vary for different consoles (i.e mobile devices with respect to TV consoles like Xbox or PlayStation). Those contextual factors can be also taken into consideration in gaming context. My research focuses on visual factors such as a visualization's background, color and size that Yao et al. categorized as contextual factors.

The field of sports analytics provides various examples of visualizations in motion falling into the aforementioned scenario of a static viewer and moving visualization. Software like FootoVision [19] and SportsDynamics [47] offer tools to embed visualizations in sport videos. For example, football matches can be enriched by visualizations regarding the kicking power of a player or the ball speed while in air. Yao et al. explored data representations in Olympics swimming competitions [65]. The authors pointed out that the main three types of representations in the context of swimming competition are symbols, text and marker lines. Those types of representations are used in video games as well. Particularly, text visualizations were moving behind the swimmers and marker line moved on the swimming pool surface to show an interpolated position of a record.

Furthermore, in the domain of Artificial Intelligence object recognition, simple visualizations are often used to label objects [34]. Usually those visualization includes rectangles, color codes, and labels with text.

Moreover, visualizations in motion with static viewers may be the result of direct interaction of the users with an interactive visualization. In fact, motion can be produced by panning, zooming and rotating the visualization its self. This case occurs frequently in video games when navigating a map in open-world games. Roth et al. [46] explored the design of interactive maps presenting a case study of spatiotemporal visual analytics of criminal activity. In their work, panning and zooming allowed to navigate on a prototyped map, creating relative motion between the map readers and the data embedded in the map.

Considering the contextual factors that could play an important role in visualization

in motion's effectiveness, I found some interesting insights in the work of Weiskopf [57]. He clarified the role of color in motion detection driving results from psycho-physical and physiological research. With the aim of providing guidelines for designing animated visualization, he suggested to use luminance contrasts for best motion perception and color patterns for grouping elements visually. In this regard, in my video game design I used colors with high contrast to increase the visibility of the visualizations embedded in the game characters.

Moreover, analyzing a second scenario proposed by Yao et al. for visualization in motion, Bezerianos and Isenberg [2] investigated the perception of visual variables on tiled high-resolution wall-sized displays. Their experiments involved static visualization and moving viewers occupied in data comparisons tasks. The first study they performed showed that accuracy was impacted by the distance between the viewer and the screen. The second study showed that perception accuracy increased with freedom of movement. Those conclusions can be related to the virtual camera moving in the game environment, facing visualizations that could be altered by camera movement, angle and distance from the data referent.

Lastly, outside the scope of data visualization, related studies on visualization in motion come from psychology. Dynamic Visual Acuity (DVA) is the ability to visually discriminate details of objects under motion [45]. Research showed that DVA depends on the contrast between the moving target and the background against which it moves. Quevedo et al. [44] highlighted that speed, trajectory, stimulus exposure time and ambient illumination influence significantly dynamic spatial resolution.

2.3. Video Games Visualization

My work is also related to past work on visualization for video games. Zammitto [67] was the first one to tackle the topic and analyze the principles of visualizations used in games. Her analysis focused on determining how video games provide the player with important visual information like the use of silhouettes, mini-maps, HUDs, Fog of war etc. Zammitto analyzed titles of three game genre identifying what kind of information are displayed in those games, which visualization techniques are used and what can be improved. Particularly, she analyzed a First-Person Shooter game (FPS), a Real-Time Strategy game (RTS) and a Massively Multiplayer Online game (MMO). She concluded that each game genre is mainly centered around tasks that require particular information and so different visualization techniques are implied. For example, health information can be displayed only with bar length or matched with color-coding. The first case works better in MMO and its usability is higher for color-blind players. The second case is

instead more common in FPS because health is the most important game element and requires high levels of details. My initial systematic analysis of the design practices for situated visualization in motion resulted in similar considerations. Each game genre used different visualization according to the main goal of the game and the current task the player is pursuing. Moreover, in my design I deviated from Zammitto consideration, using a length only encoding for character's health in the developed FPS game, since with the use of color-encoding, the game would result too easy.

Some other theories and design spaces have been proposed, based on systematic reviews of existing games that incorporate some sort of visualization. Bowman's [3] framework comprises five categories to classify any video game visualization: primary purpose, target audience, temporal usage, visual complexity and immersion/integration. Among the others, a possible primary purpose is displaying *status* information (e.g. health bars, hit direction indicators, ammo counters). Temporal usage instead, explores timing aspects of visualization uses; for example some visualization are designed to be always visible (e.g. score, power bar) while others are intermittent (e.g. in-game maps, inventory screens). Generally, the visual complexity of the visualizations embedded in games was very basic. Information should be read at a glance while the main focus of the player is on the game's primary goal. To conclude, Bowman grouped common design patterns in five categories: Heads-Up Display (HUD), Replay Theater, Progression Tree, Data in Space and Data in Time. Bowman analysis of video games visualizations helped me shape my design, taking into account the purpose of the visualization (in my case, displaying a status information such as health) and the level of integration with the data referent's itself.

Gittens and Gloumeau [20] analyzed health visual representations, exploring the impact of segmented health bars on players preferences for a game. In their pilot study, Gittens and Gloumeau designed a segmented health bar for the game *BrowserQuest* evaluating the users' preferences for the game towards the new version and the original version with a single bar with regenerative health mechanism. Their pilot proved that the majority of their players preferred segmented bar over single bar charts. I didn't use segmented bar charts instead, since I had the need to compare three different visualizations and two of the three visualizations proposed would have resulted too unusual if segmented.

Peacocke et al. [42] studied players' performances with various types of displays to find the best way to represent different types of data. Their work showed that there is no universally best display type for every information. Different representation worked well for specific types of information. As an example, they tested five different ammunition counter displays asking participants to complete 15 trials inside a custom FPS game. Result

showed that HUD options performed badly compared with diegetic¹ displays. Looking at the visual representation used in the second experiment - for health displays - Peacocke et al. explored bar charts, icons and numbers. In their third experiment about weapon displays, they also made use of color overlaying, and color-coding visualization techniques. The work of Peacocke et al. helped me design my FPS game taking into account different game mechanics possibilities such as a single room environment, no movement mechanics and custom weapons. Although, after some trial of those possibilities I decided to discard them in favour of a simpler and more linear gameplay. Also, analyzing the way they conducted the experiments on different displays, I decided to collect qualitative data instead of quantitative as they did. This decision took into consideration the available time for conducting the study, that was not sufficient to conduct a purely quantitative study.

Other than about visualizations per se, more research has been conducted in the field of video games [4, 9, 17, 23, 24]. For example, El-Nasr and Yan's [14] focused on visual attention in 3D video games considering where the user is most likely looking on the screen. Designers can utilize this research to determine where interactive elements should be placed in a game level. They found out that different game genres have different attention pattern, meaning that players concentrate on different parts of the screen according to the type of game they are playing. For example, in FPS games the players focus only on the center of the screen while in adventure games the player's gaze covers more area of the whole screen. Having this result in mind while designing my custom FPS game, I didn't place any visualization in the borders of the screen, considering always the center of the screen as primary area of focus.

Lastly, there is a lot of previous work on the use of color in video games [13, 28, 36]. Martinez et al. [36] studied color as an attribute to enrich the gaming experience, proving that color can be used to convey information in the context of video games. Specifically, they found a correlation between the type of games and the color used. In fact, I decided to use bold colors such as turquoise, light green and yellow for the ambience light of the implemented video game environment to convey excitement and the correct contrast between the visualization and the ambient colors.

¹Game elements are diegetic if they exist within and are consistent with the fiction of the game's world (e.g. the player's character perceives them and the player responds to them as such). Elements are not diegetic if they exist for the player, but not for the characters (e.g. most of the HUD or third person perspective information) [35, 42]

3 | A Systematic Review of Visualizations in Motion in Video Games

In order to survey the current practice of visualizations in motion in video games, we¹ reviewed 50 games from 17 different genres. In these games we found 160 examples of visualizations in motion that we categorize according to different dimensions. In the development of our categories we were inspired by the work of Islam et al. [26] in their analysis of existing visualizations in use on watch faces. The dimension used by Islam et al. described the visual design of data representations and the data used which we also included. We extended their categorization by adding dedicated dimensions related to motion factors and video games contextual factors. In the following sections we describe the video games selection, the categorization of the visualizations found and the results emerged from our systematic review.

3.1. Video Games Selection

To cover a diverse selection of video games in our systematic review we made use of a commercial ranking website called Metacritic [38]. Metacritic assigns a unique Metascore to each video game, which is a weighted average of the scores of the world's most respected reviewers from the gaming field. Metacritic categorizes games according to 18 different game genres. For each genre we selected the top 3 games from 2011 to 2022 sorted by the Metacritic relevance score. We considered all gaming platforms such as PC, Xbox, PlayStation etc. We excluded puzzle games from the game genres, because on a first inspection they did not contain moving visualizations. Moreover, out of the first 50 wrestling games listed on the Metacritic platform only two provided relevant visualizations. Therefore we only considered two games for these genre and not three. We excluded also

¹I was solely responsible for the data collection and analysis of the work presented in this chapter which resulted in a publication with my supervisors [6] whose text I use in this chapter. Thus any use of "we" in this chapter refers to all co-authors: Federica Bucchieri, Lijie Yao and Petra Isenberg.

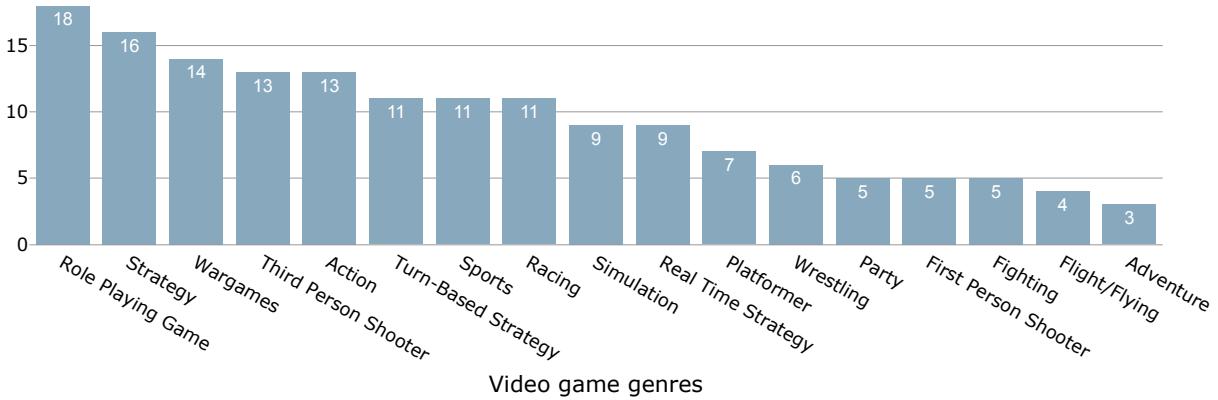


Figure 3.1: Number of occurrences of visualizations in motion per video game genre based on our systematic review.

games without a Metascore and different games from the same series (e.g *Super Mario Galaxy* and *Super Mario Galaxy 2*), due to the fact that the probability of those games providing the same type of visualization was very high. In total, we reviewed 50 games from 17 genres (see Appendix - Table A.1). For each game, I watched game-plays on YouTube for approximately 5 to 15 minutes and video-recorded relevant parts of the videos where the game showed visualizations in motion.

3.2. Categorization of Current Video Game Visualizations

In total, we collected 160 examples of visualization in motion. Among the 17 game genres analyzed, the genres that provided most visualizations proved to be RPG games with 18/160 visualizations, strategy games with 16/160 and wargames with 14/160 samples. In fact, not every genre uses a lot of visualization in motion, especially when visual representation can hinder the player's immersion. For instance, we noticed only 3/160 visualizations in Adventure games and only 4/160 in Flight/Flying games. More visualization-prone games genres are Action, TPS, Turn-Based Strategy, Racing, Sports, Simulation and RTS (see Figure 3.1). We categorized these samples according to multiple dimensions related to situated visualization and motion characteristics:

Visual representation: describes how data was represented. Signs (e.g icons, arrows) and bar charts - both linear and circular - were the most prevalent representations (36/160), followed by labels with numbers (21/160) and labels with texts (19/160). Table 3.1 summarises all the representations found.

Representation type	Occurrences	Representation type	Occurrences
Signs	36	World annotation	5
Bar chart	36	Circle	4
Label with Numbers	21	Map	4
Label with Text	19	Pictograph	4
Areas	10	Ludophasma	3
Color Overlapped	8	Heathmap	2
Silhouette	6	Pie chart	2

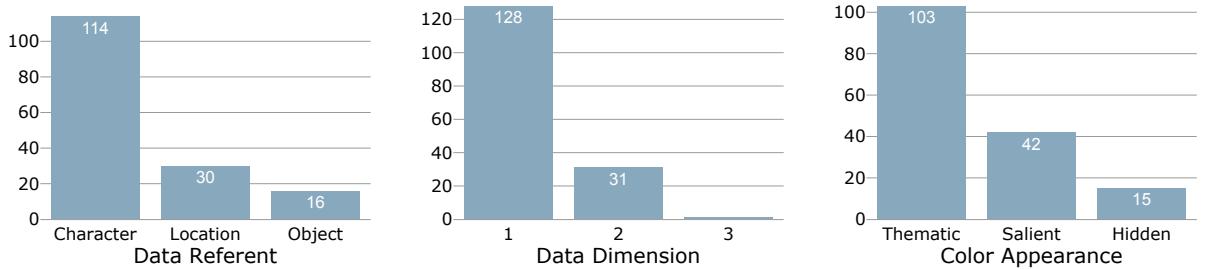
Table 3.1: Summary of the visual representations found in the 50 analyzed video games.

Data referents: are the entities that the data refers to. We found three types of referents: characters, locations, and objects. A character is an actor in the game controlled by a player or an AI (e.g. humans, beasts, creatures, robots), together with its own visible equipment like guns, armours or clothes. A location is an interactable area or point of interest inside the scope of the game (e.g. objective's position, area of attack, checkpoints). An object is any nonliving entity that the player can interact with, for example resources, potions or special powers. In our sample collection (see Figure 3.2a), ■ 114/160 data referents were characters, □ 30/160 were locations, and □ 16/160 were objects.

Data dimensions: indicates how many dimensions the visualization represents. Visualizations showing only a single information were the most common case (■ 128/160), while □ 31/160 samples encoded two types of information and only □ 1/160 encoded three types of information (see Figure 3.2b). Often additional dimensions were derived from a primary dimension. For example, while encoding the health percentage of a character in a bar chart, various games applied a color encoding scheme to represent different types of health status. A widely accepted practise was applying a traffic light color scheme varying from green to yellow to red to encode health status (healthy, medium, critical).

Color appearance: refers to the choice of colors related to the visualization's overall appearance in the game. We divided color appearance in three groups: hidden, salient or thematic. In hidden visualizations, the colors make the visualization blend into the game context. The visualization does not stand out through its design. Potentially, this makes the visualization easily ignored or difficult to read. Instead, salient colors makes the visualization highly visible. This could also be because the visualization's colors are not canonical for the type of data they show or the color combination provides a very high contrast with the game context. Finally, thematic combinations have very canonical colors for the data they represent or the color combination is well integrated with the

video game's palette. ■ 104/160's color appearances were Thematic while □ 41/160 samples were Salient and only □ 15/160 were Hidden (see Figure 3.2c).



(a) Number of occurrences of visualizations with characters, locations and objects as data referents based on our systematic review.

(b) Number of occurrences of visualizations with 1, 2 or 3 data dimensions based on our systematic review.

(c) Number of occurrences of visualization with a hidden, salient or thematic color appearance based on our systematic review.

Figure 3.2: Results of the systematic review categorization for three dimensions: (a) Data referent, (b) Data Dimensions and (c) Color appearance.

Background consistency: considers if the visualization has a fixed background or the background varies according to the visualization's position. When enemies move from one scenario to another (i.e. different rooms in a closed game environment, different landscapes in an open game environment), the visualizations attached to them will be displayed on the changing backgrounds. If the background of a visualization is fixed, for example providing an opaque container behind it, there is a higher possibility that the visualization is always visible regardless of motion factors. Indeed, visualization colors can be difficult to distinguish on some backgrounds compared to others. Results show that ■ 133/160 visualization's backgrounds were varying with the visualization position, while only □ 27/160 had a fixed background (see Figure 3.4a).

Embedding locations: refers to the spatial relationship of the visualization and the data referent. We found three different types of embedding locations (see Figure 3.4b). ■ 121/160 examples embedded visualizations *around the data referent*. Visualizations were close to the referent for example above an object (see Figure 3.3a)), under the feet of a character, or above a checkpoint (a location). □ 26/160 visualizations showed full or partial *overlap with the data referent* as shown for example in Figure 3.3b. Finally, □ 13/160 visualizations were *integrated with the data referent* permanently for example in the material color of the referent, in its design or in the diegetic elements attached to it (i.e. equipment for characters as Figure 3.3c shows). Although visualizations integrated or overlapped with the referent might look similar, integrated visualizations cannot be



(a) Stamina bars and names embedded below the feet of the players in *NBA 2K21*.

(b) Health map overlapping the game environment in *Civilization VI*.

(c) Health counter integrated with the character's suite design in *Nintendo Land*.

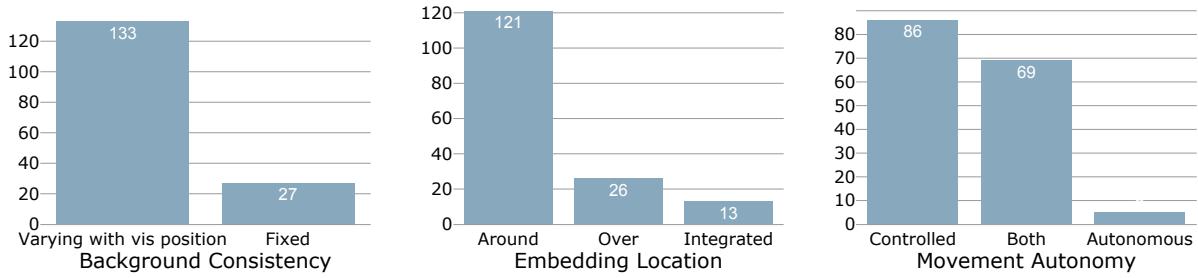
Figure 3.3: Examples of visualizations in motion embedded in different locations with respect to the data referents.

separated from the referent while overlays may be temporary.

Movement autonomy: considers if the visualization was moving autonomously or was controlled by a player. For example, in Third Person Shooter games (TPS), the player is controlling a main character defined as the protagonist. By moving the protagonist, rotating its view and changing the camera perspective, the player can induce motion of other elements on the screen. Autonomous movement instead, does not depend on the player's control for example when the camera is still and some characters are moving on screen. Autonomous movement was not prevalent in video games, only 5/160 samples moved autonomously, while 86/160 were consistently controlled in some way by the player and 69/160 depended on both autonomous movement and player's control (see Figure 3.4c).

Visibility conditions: considers if the visualization is only visible when the player initiates visibility by focusing on the data referent. For example with hover, by pushing a button, using a special power and more. In this case we consider those visualizations to be *on focus*. If they are visible independently of the player's actions, they are *not on focus*. This mechanism is very common in First Person Shooter (FPS) games where enemy health bars are only visible while the cross hair of the protagonist hovers the body of the enemies themselves. The slight majority of the cases were not On Focus visualization (92/160), while 68/160 visualizations were On Focus (see Figure 3.5a).

Frequency of appearance: this dimension is intended to evaluate how frequently a visualization in motion appears during a normal gaming session. Since frequency is a relative measure, we categorized samples in four categories (see Figure 3.5b): Rare ()



(a) Number of occurrences of visualization with a background varying with the visualization's position or with a fixed background based on our systematic review..

(b) Number of occurrences of visualization embedded around the referent, overlapping the referent or integrated in the referent's design based on our systematic review.

(c) Number of occurrences of visualizations moving autonomously, prompted by the player's control or depending on both factors based on our systematic review.

Figure 3.4: Results of the systematic review categorization for three dimensions: (a) Background Consistency, (b) Embedding Locations and (c) Movement autonomy.

11/160) were those visualizations that appeared in exceptional cases (e.g. progress bars of special actions); Moderate (■ 13/160), defined visualizations that appeared from time to time (e.g. secondary tasks related visualizations such as troops movement in resource management games); Frequent (■ 118/160), appeared often but not contiguously (e.g. primary task related visualizations such health bars on focus for FPS games); Always Visible (□ 18/160) were those visualizations that were always displayed on screen (e.g. health of the main character in TPS game).

Foreground: considers if the visualization is always in the foreground and cannot be overlapped. Occlusion is a very important factor to consider in video games. If we imagine a scene with 20 enemies in front of the game protagonist, each one of them with a visualization displayed under their feet, it is likely that some of the visualizations will be poorly visible or totally hidden due to the occlusion caused by other elements. Also environment elements can occlude visualizations. For example, a tree can hide some visualizations displayed near by. Being always in the foreground can help the visualization stand out and be more readable. In 110/160 ■ cases visualizations were always in the foreground, while in 50/160 □ cases the visualization could be overlapped by other elements in the game (see Figure 3.5c).

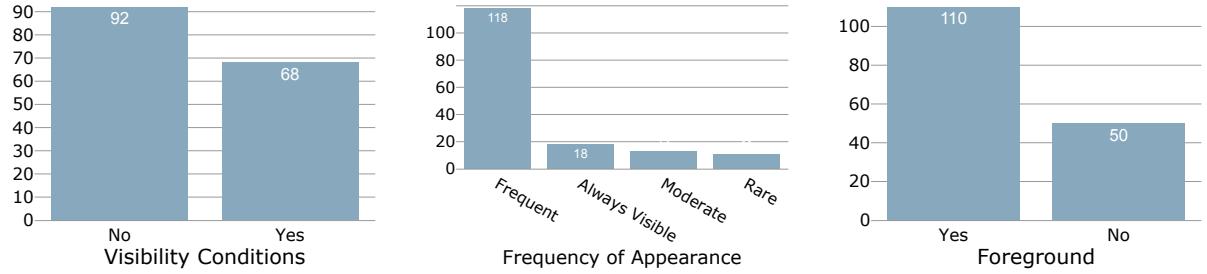


Figure 3.5: Results of the systematic review categorization for three dimensions: (a) Frequency of appearance, (b) Visibility conditions and (c) Foreground.

3.3. Relationships between Different Dimensions

The design of visualizations in motion for video games comes with some constraints. First and foremost the video game genre. Visualization for different game genres may vary due to the game context they have to fit in. Looking at Figure A.10 (see Appendix A), some visual representations appear to be used only in some game genres. This is the case of *heatmaps*, found only in Turn-Based Strategy (TBS) games or *pie charts*, present only in Real Time Strategy (RTS) and wrestling games. Other form of visual representation instead proved to be widely used independently of the game genre. We found *signs* in 15/17 game genres, *bar charts* in 11/17 genres, *labels with text* in 12/17 genres and *labels with numbers* in 9/17 genres.

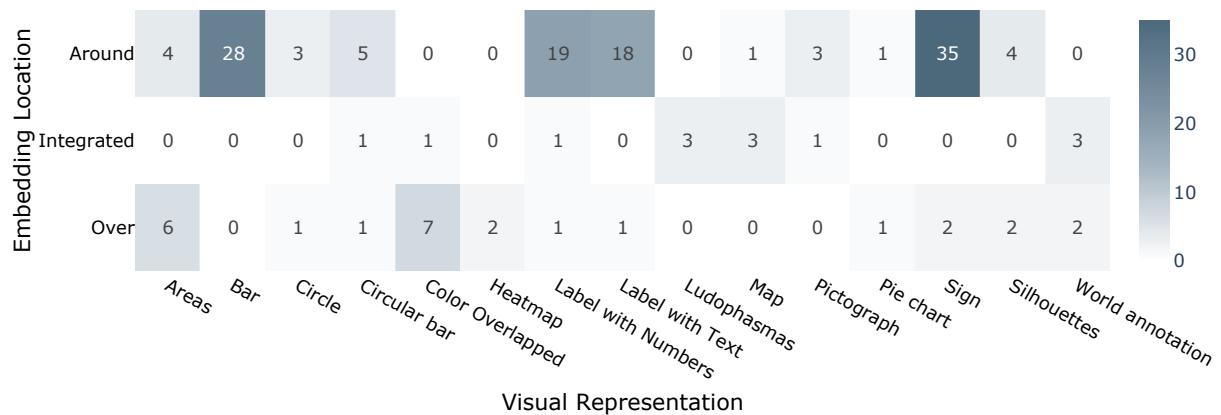


Figure 3.6: Number of occurrences of different visual representation placed in each embedding location emerged from our systematic review.

The game genre influences also the use of color in the visualization design, thus the visualization's color appearance can depend on the game genre. Figure 3.7 shows that *salient* color combinations are mainly used in Role Playing Games (RPG), flight/flying, war and wrestling games. Adventure and platformer games, instead, use only *thematic* color combinations. Moreover, other genres prefer thematic color appearance of their visualizations. This is the case of strategy, sports, racing and simulation games.

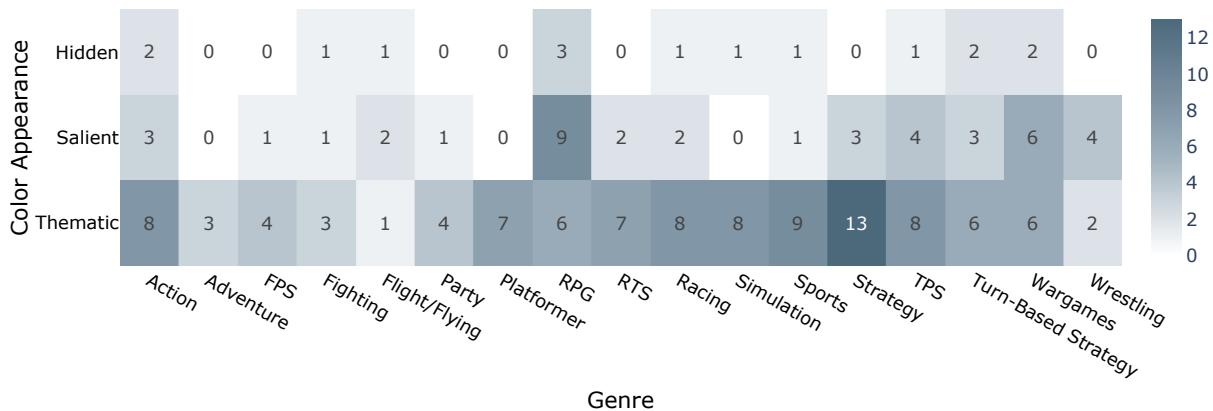


Figure 3.7: Number of occurrences of visualizations with different color appearances in different game genres, based on our systematic review.

Another constraint in the visualization design is the embedding location the visualization should be placed in. For example, visualizations placed below the feet of characters are frequently subject to occlusion problems due to the character's legs. In this cases, visualizations like heatmaps, pie charts or labels with text can be less readable. According to our sample collection (see Figure 3.6), some visualizations are designed to be displayed in only one of the three embedding locations identified. Bar charts are embedded only *around the data referent*, as well as most of the signs, labels with text and labels with numbers. In contrast, maps, ludophasmas - in game sprites that usually represent player ghosts or cloned players, used to show historical data (i.e. replays) of the gameplay [37] - and world annotations are *integrated in the referent design* in the majority of the samples collected. Lastly, visual representations such as areas, color overlapping and heatmaps typically *overlap the data referent*.

Analysing the impact of the embedding location in the visualization design, we can explore the relationship between the embedding locations and the visibility conditions in which a visualization is shown (see Figure A.9). The majority of the visualizations embedded around the data referent or integrated with its design are visible without the need of initializing visibility action (*not on focus*). On the other hand, visualizations overlapping the referent are frequently displayed *on focus*.

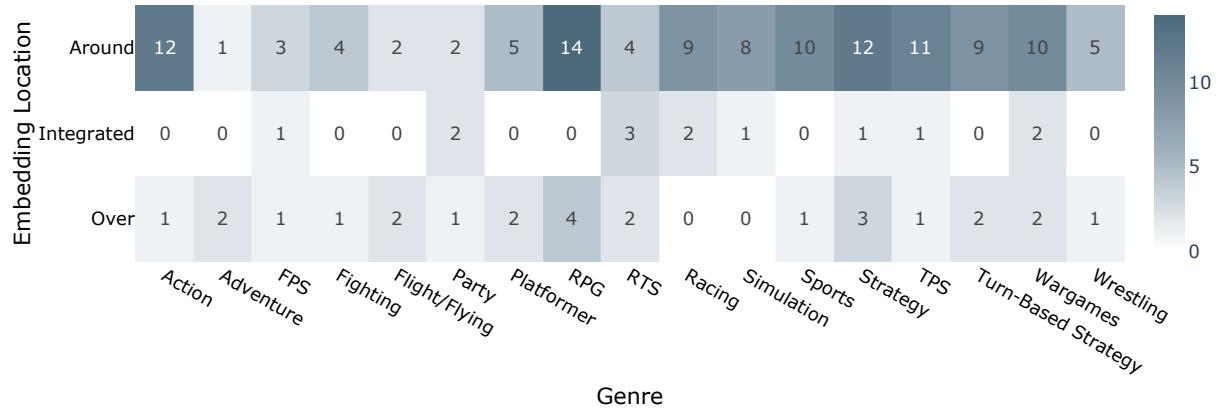


Figure 3.8: Number of occurrences of visualizations embedded in the referent in different locations in different game genres, based on our systematic review.

Lastly, exploring the movement autonomy of different data referents, we can clearly notice that it is rare for locations and objects to move *autonomously* (see Figure A.8). The motion factor in those cases should be triggered by other factors (e.g. a character throwing an object on the ground). Also, we can see that locations and objects are mostly static entities by the fact that the majority of their movements are *controlled* by the player actions. Regarding characters instead, their movement autonomy relies mainly on *both factors* - player's control and autonomous movement (e.g. health bar above the head of an enemy moving because of the enemy's locomotion and on the main character movement).

4 | Analysis of different types of data in video games

After conducting the systematic review of visualization in motion, we¹ specifically concentrated on how different types of data are represented in video games. Video games produce a wide range of types of data. A single visualization, depending on the type of encoding used, can represent multiple types of data. For example a health bar using length and color represents both quantitative data (the health points of a character) and categorical data (the character's team affiliation). An icon to represent the target of a game mission instead, can encode both spatial data (the distance of the protagonist from the target) and categorical data (the type of target, for example a place or a person). □ 67 out of the 160 visualizations we analyzed represented *quantitative data* (see Figure 4.1a). This data concerned health points of a character, its stamina, or the number of resources crafted. *Categorical data* was the second most common category, with □ 64 out of 160 representatives. Categorical data concerned team identification and resource type. Furthermore, □ 49 out of the 160 visualizations showed *spatial data* for the data referent's position. Meanwhile, *ordered data* was the least frequent with □ 8 out of 160 representatives and was mainly present in racing games (i.e. the position of the opponents in the leader board). In fact, some game genres were more prone to represent specific types of data with respect to others (see Figure 4.2). Quantitative data is very popular in strategy games, RPG and war games. Instead, categorical data is frequently embedded in action, racing, sport and war games. Spatial data are used in TPS games, action games and RPG. Lastly, ordered data is only used in some specific game genres like racing, simulation and TPS games. In the next section, we analyse how different types of data are represented in video games. In section 4.2 and section 4.3 we focus on the two most frequent quantitative and categorical data displays: character's health and game element types.

¹I was solely responsible for the data collection and analysis of the work presented in this chapter which resulted in a publication with my supervisors [5] whose text I use in this chapter. Thus any use of "we" in this chapter refers to all co-authors: Federica Buccieri, Lijie Yao and Petra Isenberg.



Figure 4.1: Results of the analysis of different types of data in video games.

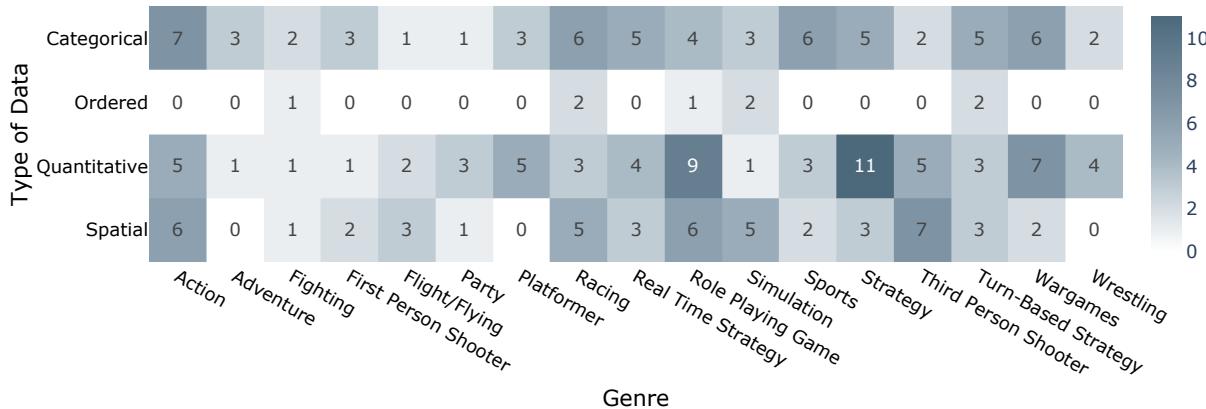


Figure 4.2: Number of occurrences of visualizations displaying different types of data in different game genres based on our systematic review.

4.1. How different Types of Data are represented in Video Games

As emerged from our sample collection (see Figure 4.3), *quantitative data* was mainly represented by means of linear bar charts (■ 28/67), labels with numbers (□ 14/67) or circular bar charts (□ 7/67). Instead, *categorical data* was represented as labels with text (□ 18/64), signs (□ 17/64) or by color-overlapping on the data referent (□ 7/64). Looking at how spatial data was represented, the vast majority of the examples found used signs (■ 26/49). This representation was very common for mission's objectives or interactive items position. Also the use of world annotations² was canonical for spatial

²world annotations: visual representations added to the game world, such as the optimal racing line in driving games and highlighted objects in the scenario. [3].

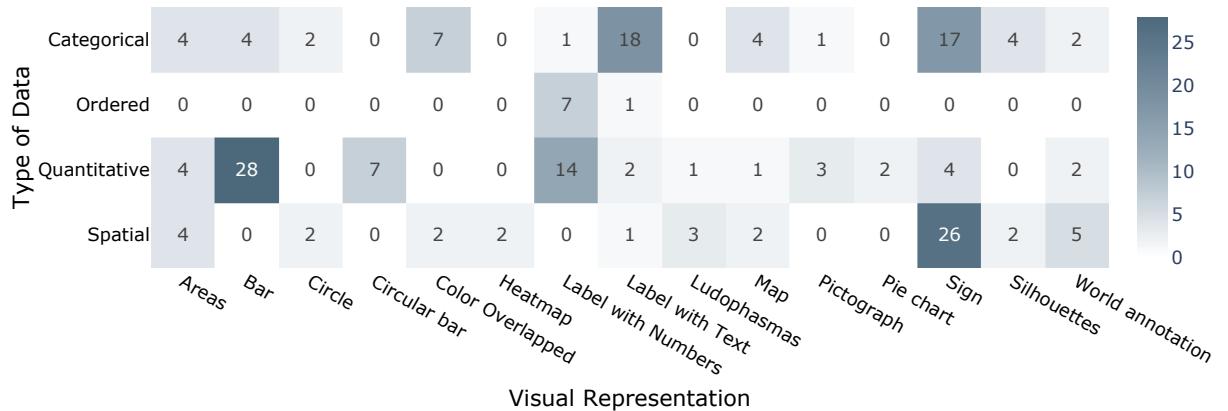


Figure 4.3: Number of occurrences of visualizations with various visual representation according to different types of data, based on our systematic review.

data representations (◻ 5/49). Lastly, *ordered data* was represented with labels both using numbers (■ 7/8) or text (◻ 1/8). Analyzing the spatial relationship of the visualization and the data referent, there were some tendencies in the way different types of data was *embedded* in the data referent (see Figure 4.1b). In our sample collection, the vast majority of each type of data was embedded around the data referent. Moreover, ordered data was only displayed around the data referent, without any sample of visualization overlapping or integrated with the data referent. For others types of data, every embedding location included at least some samples. Looking at *spatial data*, ■ 29/49 visualizations were embedded around the data referent, ■ 12/49 were overlapping the referent and ◻ 8/49 were integrated in the referent's design. Lastly, regarding the movement autonomy of different types of data, there is a strong tendency for visualization to be in motion due to the player's control (see Figure 4.1c). Ordered data was the only type of data that showed a majority of samples with a movement autonomy depending on both controlled motion and autonomous motion (■ 7/8). Quantitative, categorical and spatial data had mostly a controlled movement autonomy with just few cases of autonomously moving visualizations.

4.2. Character's Health

While designing video games, the concept of health is prominent and important and that is why I dedicate a separate section to it. Health can be defined as an attribute of a game element that determines the maximum amount of damage that the element can take before dying or being destroyed [62]. Previous research explored the concept and the visual representation of this attribute. Fabricatore et al. [17] proposed a game design reference mirroring players' preference. In their qualitative model the concept of health is

generalized as *energy*, defined as a resource that allows an entity to exist in the gaming world. Changes of energy are a direct consequence of losses and increases. Living game entities, such as characters, can lose energy by fighting against opponents, falling from high heights or interacting with harmful game elements. Inanimate entities, such as objects or vehicles, can lose energy if damaged by third parties or as the result of a collision with other entities. Clarke and Duimering [9] conducted an exploratory interview-based study of computer gaming and reported interesting results about First Person Shooter (FPS) game genre. By interviewing players, Clarke and Duimering reported the gamers' need to have an enduring on screen representation of health. As a matter of fact, health is considered an essential feature in video game design. Brooksby [4] in his exploratory study about representation of health in video games, investigated further representations of health in well-known video games and identified five categories of health representation: mobility, ability, psychology, social and pain. For example mobility health related representations showed the inability of a character to move, while pain depicted a damage or an injury. Unsurprisingly, the most frequent depiction of health appeared to be pain. This category of health related representation is, according to Brooskby, well embedded in game design.



(a) Bar chart in *Aliens: Fireteam Elite*.



(b) Radial bar chart in *The Falconeer*.



(c) Pictorial fraction chart in *Skies of Fury DX*.

Figure 4.4: Different types of character's health visualizations.

4.2.1. Analysis of character's health representation

According to the literature, health is usually represented in the form of an exact numerical value or percentage. Graphically it can be represented by a bar chart [4][3] or numbers [29][67]. Looking at the samples collected during the systematic review, out of 67 *quantitative data* visualizations, 24 represented the character's health. Fabricatore et al.'s definition of health was supported by our findings, as well as the player's need for health indicators point out by Clarke and Duimering. The most used representations of health were horizontal bar charts (■ 18/24 visualizations; see Fig. 4.4a). Only □ 3/24 representatives were radial bar charts (see Fig. 4.4b), and the remaining □ 3

were a pie chart, a label with numbers, and a pictorial fraction chart (see Fig. 4.4c). Considering the encoding used, health was mainly represented with single-color horizontal bar charts that encoded data only by bar length (■ 19/24). □ 3/24 examples showed a dual encoding with the use of color, usually in gradients from green (healthy) to red (critical). The remaining □ 2 samples out of the 24 encoded health by angle plus color encoding or only by using numbers. Regarding the background consistency, ■ 16/24 visualizations in our collection had an opaque background, while the remaining □ 8 had a transparent background. Having a transparent background also meant that these bars lacked a reference frame that could help to judge the length of the bar more precisely [10]. Also, from the data collection emerged that the vast majority of the character's health visualizations were embedded around the data referent (■ 22/24). In fact, only □ 1 visualization was overlapping the data referent and only □ 1 was integrated in the character's design. To conclude, we evaluated also the visibility of health visualizations. We discovered that ■ 15/24 were visible only on focus so needed the player to initiate visibility by focusing on the data referent. The remaining □ 9/24 health visualization were always visible, independently from the player's control.

4.3. Game Element Types

To succeed in certain games, finding the right resource or correctly identifying enemies is important. This type of information is usually represented by *categorical data*. By game elements, we mean interactive objects, characters, locations and more. Hoobler et al. [23] analyzing competitive behaviour in multi-user virtual environments, proposed an example of how characters from different teams are represented. When looking at the position of players, it is important to understand where, for example, your allies are located. Hoobler et al. proposed player glyphs to visualize team identification, using color encoding. In their game (*Wolfenstein: Enemy Territory*) two teams are facing each other, the blue and the red team. Particularly, they used icons of two different colors to convey the team affiliation of each player. Moreover, Hoobler et al. pointed out that in many objective-based team games, players are divided into classes. Different classes are associated with different skills and abilities and it is important during gameplay to identify who is capable of what. In their player's glyphs, they used signs to distinguish, for example, medics from engineers. Horbiński and Zagata [24] focused on symbols interpretation on the map of the *Valheim* video game. Inside the game map, different interactive elements (mainly locations and objects) are presented by pictographs to inform users about the game. Since *Valheim* is a survival game, understanding the meaning of each symbol is crucial for the players. In his dissertation on information visualization in games, Karlsson [29] states how important

colors are used to distinguish enemies from allies in First Person Shooter (FPS) games and Massively Multiplayer Online Role Playing Games (MMORPG). Specifically, he analyzed *Battlefield 4* and *World of Warcraft (WOW)*. The results of his experiment shows that, while playing *WOW*, players knew they were facing an enemy because their name and health bar was colored in red. Likewise, in *Battlefield* color was used to present the status of flags to conquer by means of a status bar color. The bar was colored in red when the enemies were capturing a flag or in blue when allies were capturing it.

4.3.1. Analysis of game element types representation

We found 26/160 visualizations that represented the type of game elements, such as interactive objects and characters. Signs were the most common visual representation, with \blacksquare 13/26 visualizations. Signs used shape and color encoding to distinguish different categories of game elements, such as mission targets (see Fig. 4.5a). Those encodings aimed to help players differentiate between other players, objects, and objectives [29]. Another method to identify character types is the use of color in the character's health bar charts (\blacksquare 5/26, see Fig. 4.5b). \blacksquare 3/26 elements in our sample collection were Silhouettes, representing the type of game element by a colored outline. The remaining elements (5/26) present 5 other different visualizations that were not considered relevant due to the low number of examples found. When categorizing the 26 visualizations found by the type of encoding used, it is possible to notice that the majority used color-only with \blacksquare 12 out of 26 representatives, while \blacksquare 8 out of 26 visualizations used both shape and color, and \blacksquare 5 used only shape (see Fig. 4.5c). The last representative (\blacksquare 1/26) displayed information by using text. Differently from the character's health representation, there is no strong design preference concerning the background of visualizations concerning types of game elements. \blacksquare 13/26 visualization had a fixed background and \blacksquare 13/26 had a transparent background. To conclude, also in this case the majority of the visualizations were embedded around the data referent (\blacksquare 21/26), only \blacksquare 3/26 overlapped the data referent and only \blacksquare 1/26 was integrated with its design.

4.4. Summary

Our results showed that quantitative and categorical data were the most recurrent types of data visualized in video games. We observed a correlation between the video game genre and the types of data displayed in the game. According to our sample, quantitative data was mainly represented by bar charts and labels with numbers. Instead, categorical data was represented by labels with text or signs. Moreover, we evinced a tendency for different



(a) Missions' icons in *Spider-Man*. (b) Color coded health bar charts in *League of Legends*. (c) Resource's type signs in *Aliens: Fireteam Elite*.

Figure 4.5: Different game element's type visualizations.

types of data to be embedded in a specific embedding location but embedding data around the data referent was the most used embedding location for any type of data. Lastly, quantitative, categorical and spatial data visualizations mainly moved due to the player's control, while ordered data also was often associated with the autonomous movement of the data referent.

We then analyzed two prominent examples of data displayed in games: character health and game element types. Character health (quantitative data) was predominantly represented by bar charts using a length only encoding, opaque backgrounds and the visualization were mainly embedded around the character. Instead game element types (categorical data) were mainly visualized by using signs and color encoding. As for the character's health, game element types visualizations were also mostly embedded around the data referent.

5

RobotLife: A Video game for the Evaluation of Visualization in Motion

With the aim of exploring how different contextual factors influence visualizations in motion in the context of video games, I designed and implemented a video game called *RobotLife*. The goal was to exploit the data produced by the gameplay and visualize it to study the impact of contextual factors on situated visualizations embedded in game elements subject to motion. To study the impact of motion, there was the need for relative motion factors of adequate magnitude between a static player and the visualizations displayed on screen. Moreover, the game scenario and mechanics needed to be centered around the visualizations themselves. In this way the effect of motion on the visualizations could impact the overall game performances and experience of the player. *RobotLife* is a First Person Shooter (FPS) game set inside a robot factory. The player takes on the role of the factory's guardian and has to kill robots that want to destroy the factory. In this chapter I describe the video game design and details, the characters of the game and the game mechanics, motivating all the design choices behind the implementation of *RobotLife*.

5.1. First Person Shooter games

The first step of my video game design journey was choosing the game genre to implement. Since the impact of relative motion between the player and the visualizations depends on the type and magnitude of the relative motion itself [66], choosing a motion-prone game genre was crucial to guarantee a meaningful impact of motion on the visualizations' readability. From the video game genres analyzed during the systematic review, the First Person Shooter (FPS) was selected as a motion-prone game genre. FPS [12, 15] can be defined as a sub genre of action games where the player experiences the game from a first person perspective and has to face enemies of different natures (other players or AI game

(a) Screenshot from *Call of Duty: Black Ops 4*.(b) Screenshot from *Cyberpunk 2077*.

Figure 5.1: Examples of first-person perspective in FPS games, showing the hand of the protagonist holding a weapon, navigating a 3D environment and facing different types of enemies.

characters) while navigating a 3D environment and completing some game missions (i.e. kill all the enemies, reach a specified location, rescue an object). Looking at Figure 3.1 the decision to chose FPS instead of genres that produced more examples of visualization in motion may seem inconsistent. However, a single video game can fall under different categories. Discussing the list of selected video games presented in the Appendix A - Table A.1, the game *Call of Duty: Black Ops 4* (Figure 5.1a) was categorized as an action game. Still, *Call of Duty: Black Ops 4* falls legitimately in the definition of FPS game according to Metacritic's categorization [39]. Analyzing *Cyberpunk 2077* (Figure 5.1b), categorized as Role Playing Game (RPG) by Metacritics [40], it is possible to notice that the game presents a first person camera perspective and shooting is one of the main actions required in the game. The popularity of the FPS game genre guided also the decision process. According to the New York Times [32], back in 2013 FPS video games already monopolized the video game market, with more than 2 billion dollars of income per year. Moreover, the body of research on FPS games is extensive [1, 11, 18, 25, 27, 33, 55].

5.2. Implementation Environment

RobotLife was implemented using Unity, a cross-platform game engine developed by Unity Technologies [50]. Due to the time constraints of this project and the average time required to implement a video game, I decided to start my implementation from an open-source FPS video game. My research of possible open-source games focused on finding a game with the possibility to manage data produced by the gameplay and embed visualizations with the game elements. Among the explored possibilities, I selected the *FPS Microgame Template* by Unity Learn [48] (see Figure 5.2). *FPS Microgame Template* is a 3D First Person

Shooter game available for free directly on the Unity Learn platform. Game developers can modify and customize the game by following some lessons provided by Unity Learn or according to their creativity. The microgame comprises level building assets, weapons, props, enemies, and more. Going more in details, the microgame comes with the basic game mechanics already implemented: a first person character controller provides an input recognition mechanism, a rudimentary environment is proposed in the sample scene, as well the shooting mechanics to fight the enemies in the scenario. The final goal of the Unity Learn Microgame is to kill all the enemies in the game scenario.

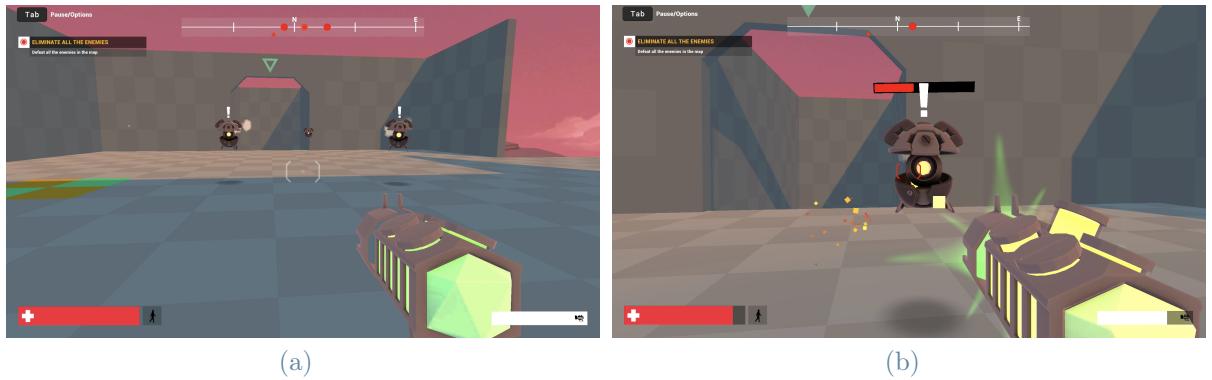


Figure 5.2: Screenshot from the *FPS Microgame* Template by Unity Learn. No modifications were applied to the microgame at this stage.

5.3. RobotLife Storyline

RobotLife is set inside a robot factory. A virus infiltrated inside the factory and altered some robots. The altered robots became evil and started damaging the electric system of the factory. The player acts as the guardian of the factory and has the duty to eliminate all the evil robots. This task comes with a main challenge: distinguishing the evil robots from the good ones. Indeed, the evil robots are acting secretly, trying not to be recognized while destroying the factory. The virus gave extra health to the evil robots so they come with more health with respect to the good robots. This is the only distinction between good and evil robots.

Usually, the robots are not dangerous for humans, but every robot comes with a life-saving mechanism so when attacked, they fire back immediately. The central management system of the factory spotted 8 evil robots connected to the electrical system of the facility. The player wins the game when all the 8 evil robots are eliminated and not more than 2 good robots were damaged.

5.4. RobotLife Characters: The Hoverbots

The FPS Microgame proposed two types of enemies prefabs¹: the hover bots and the turrets. The most interesting one for my research was the hover bot (see Figure 5.2). Hover bots are flying robots of middle dimensions able to shoot at the player and follow him or her fluctuating inside the game scenario. The hover bot features ensured the presence of autonomous motion factors in the game, other than movements caused by the player control. The structure of the hover bots comprises three parts: a bottom part, an eye-ball and a top part similar to an helmet. The body of those robots is colored in grey with a predefined texture applied on it. The eye-ball instead as an iconic yellow color and it represents also the spawning point of the projectiles fired by the hover bots. The robots are subject to animations that create a realistic and funny characterization of those enemies. For example when moving around, the top part of the robots has an inclination of 45 degrees to look more aerodynamic, occluding partially the eye-ball back side. Also, the idle animation makes them bouncing smoothly in their current location. Moreover, hover bots comes with easily accessible data such as their position in the scenario, their status (idle, patrolling, following and attacking) and their health level. Exploiting the knowledge acquired during the analysis presented in chapter 4 I decided to focus more on the health level. The decision to use the health level as main piece of information to embed in the hover bot is motivated in the following chapter.

5.5. Game Specifics

My main goal while designing RobotLife was to create specific game mechanics that would push users to mainly focus on the situated visualizations inside the game, while maintaining the classical gaming experience of a FPS game. As pointed out in section 5.1, the main characteristics of a FPS are: the first person camera perspective, the navigation inside a 3D environment, the use of weapons and the presence of enemies to defeat. Therefore, the proposed storyline is centered around fighting the hover bots. After deciding that the data I wanted to visualize was the health level of the hover bots, I decided to build the main objective of the game around this piece of information. Therefore, I implemented the division of the enemies in two "teams", good and evil robots. The division is made considering the starting health level of the robots. Good robots have a health level lower than 66% while evil robots have a health level higher than 66%. This division allowed to force the study's participants to focus on visualizations of the health level. Furthermore,

¹A Prefab is a stored GameObject complete with all its components, properties, and child GameObjects that could be used as a template to create new Prefab instances at any time inside the Unity Scene.



Figure 5.3: Screenshot of RobotLife UI. The player’s health is displayed on the bottom left part, using a bar chart in red and white cross symbol. The ammunition count is displayed on the bottom right part, using a bar chart in white with a weapon symbol. Primary and secondary objectives of the game are displayed in the left part of the screen, together with the pause button.

the rule of not-damaging the good robots was intended to avoid participants shooting blindly at every robot having in mind only the main goal of killing the 8 evil robots in the scenario. In fact, damaging more than 2 good robots causes an immediate game over.

RobotLife is intended to be used only to study situated visualization in motion in a real video game scenario. For these reason the game storyline was kept simple and only one level is available inside the game. Moreover the level map and game aesthetics should reach the standards of a real video game. The level map was designed to be linear but at the same time challenging and big enough to support navigation without information overload. Some areas of the map encourage players to jump between platforms or over decorative objects, with the aim of creating extra motion factors in the gameplay. Moreover, the map was designed to be easy to navigate. In fact, some doors were placed between different rooms with the purpose of reducing the difficulty of navigation and allow to switch from one room to the others as easier as possible. Robotlife has a cartoon-style graphic, with a Sci-Fi aesthetics design, with Led lights colors on the tons of green, blue, cyan and violet.

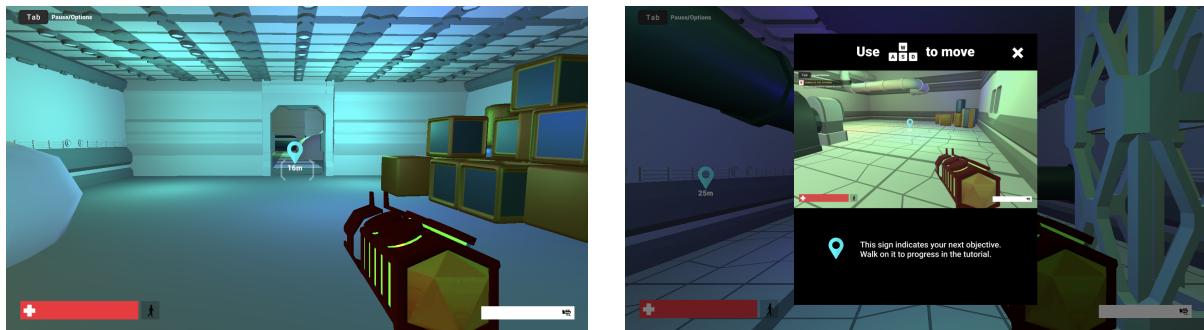
Finally, the RobotLife UI was kept minimal and essential, as Figure 5.3 shows. On the bottom part of the screen the health level of the player is displayed on the left, while the ammunition count is on the right. Both health and ammunition count are represented as a single bar chart with two symbols on each bar to convey what they are representing: a

34 5 | RobotLife: A Video game for the Evaluation of Visualization in Motion

cross for the health and a weapon for the ammunition. A classical crosshair is displayed at the center of the screen to help the players aiming at the robots. When the player targets a robot, the crosshair turns red providing responsive visual feedbacks. On the left part of the screen the primary and secondary objective of the game are displayed. The primary objective is killing evil robots and it is presented with a red dot icon. The secondary objective is not to damage good robots and it is presented with a light blue icon. Also the order in which the objectives are presented conveys the hierarchy of the objectives. Lastly, a pause/options button is displayed above the objectives.

5.6. Tutorial

Before starting the main game level there is a tutorial level that explains the commands of the game and the main game mechanics. Each player is encouraged by the game to follow the instructions and to take all the time needed to complete the tutorial. In case of necessity, the tutorial can be repeated more than one time. The tutorial has various waypoints as objectives: intermediate points in the game world placed between the starting position of the player and the tutorial's exit point. The waypoints are ordered and the player is guided through each of them thanks to a waypoint indicator, as shown in Figure 5.4a. Figure 5.4b shows the instruction panels displayed when the player reaches specific waypoints that introduced new game commands. In those panels, new commands are explained with detailed screenshots to help the player comprehend and learn how to play the game by playing directly. Particularly, the tutorial teaches how to move in the game world, to shoot and kill enemies, to gather extra health, to aim and to jump.



(a) Screenshot showing the next waypoint indicator and its distance to the player.

(b) Screenshot showing the first instruction panel about how to move in the game.

Figure 5.4: Screenshots of the tutorial level of RobotLife.

6 | A Study Design for Visualization in Motion in Games

As one of the main contributions of my thesis, I propose a study design for situated visualization in motion in video games. The study aimed at evaluating how different embedding locations affects the readability of visualizations in motion in a real video game scenario, how different embedding locations impact the aesthetics of the game characters and the overall experience of the players. The study explored motion factors in the context of static viewer and moving visualization [66]. In this section I motivate the decision of studying the embedding locations as main contextual factor, I present the study variables, the game flow, and the visualization designs evaluated.

6.1. Contextual Factor to Evaluate

Studying the impact of different embedding locations seemed the perfect match between situated visualization and visualization in motion characteristics. The visualization's embedding location can have a direct impact on the visualization visibility, background, color contrast and much more. For this reason, embedding a visualization in different locations can have a direct impact on the visualization's readability and on the data referent's aesthetics. For example, displaying a visualization around the character automatically increases the visibility of the character itself, providing an indicator of its position. Moreover, embedding a visualization in the character design impacts the character's aesthetic by altering its original design. Visualizations overlapping the character are instead more prone to occlusion due to other objects in the game scenario.

Analyzing other possible contextual factors to study, I excluded several of them basing my exclusions on some criteria depending on the FPS Microgame design and on previous works about video game visualizations. Starting from the FPS Microgame design, the hover bots come with some constraints. First of all, those characters fluctuates around

the scenario. Considering the player navigating the game environment, the movement autonomy of the embedded visualizations is based on both the robot's movement and the player's control. So I excluded *movement autonomy* from the factors worthy to study due to robot's design constraints. Secondly, choosing a game character as referent excluded *data referent* from the list of possible factors to study. Moreover, health is usually an information displayed frequently in FPS so, by design, every robot has to display its health level at any moment, excluding the *frequency of appearance* from the list. Moreover, since health is quantitative data usually represented as a numeric value from 0 to 100, I decided to consider health directly as a percentage as in the original game design. Looking instead at the previous work done in the field of video game visualizations I excluded *visual representation* because it was similar to what Peacocke et al. [42] studied. They investigated different information displays in FPS games, concerning different tasks: monitoring health, monitoring ammunition, matching weapons to the situation and navigating the environment. Looking at the monitoring health task, the different information displays can be compared to different visual representations embedded in the characters.

6.2. Choice of Data to Embedded in the Characters

As shortly introduced in section 5.4, I selected the hover bot health level as the piece of information to visualize and embed in game characters. The hover bots health goes from 0% to 100% and it was originally represented with a red bar chart embedded above the head of the robots. It was possible to set the bar chart as always visible or as initially hidden. When hidden the bar chart became visible only when the player shoots at the robots for the first time. From that moment on, the bar was always visible and rotated around a pivot point to face the camera/player at any time. The bar chart was implemented using two 2D images, one black that was used as background and remains static, and a second one placed on the foreground that was instead red. The red image varies its length to represent the current health value of the hover bot it was embedded with, thanks to a dedicate script attached to the game character element.

In real FPS video games, the health level is a critical information that could determine the gaming strategy a player would like to adopt to win the game. For example, while facing an enemy with low health, the player is conscious that he or she has only to hit the enemy a small amount of times before defeating it. Instead while facing an enemy with full health, the player will maybe be more careful and shoot more time to hit the target as much as possible. It is also important to notice that the usual game mechanics of an FPS

video game are centered around killing all the enemies without distinction. In this case, the player does not need to read the exact value of the health representation, but just to evaluate how close the value is to 0. The player final goal is always to bring the health value to 0. For this reason, RobotLife game mechanics were implemented in a different way, forcing the players to actually read the health level and not blindly shoot at every enemy in the scenario.

6.3. Choice of Visualizations to Evaluate

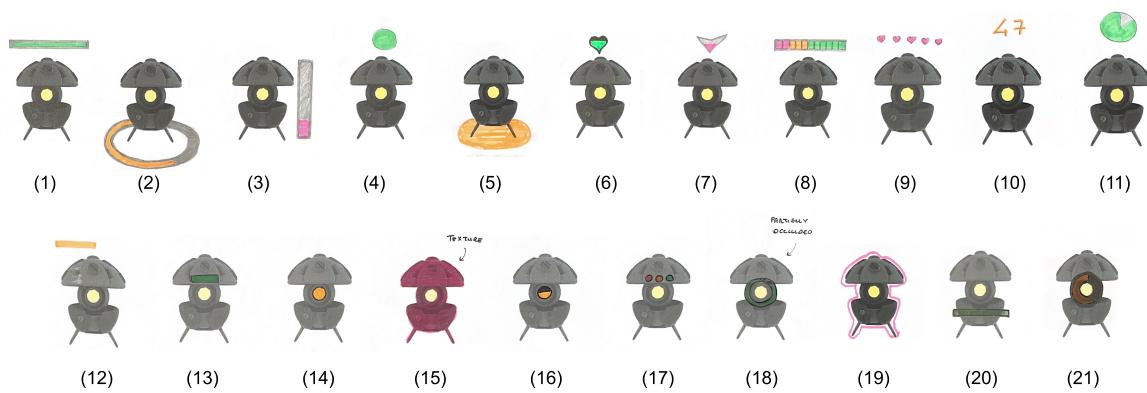


Figure 6.1: Visualizations sketched during the brainstorming phase presenting different visual representations, embedding locations, colors and types of encoding.

During an initial brainstorming phase, I sketched various possible health visualizations embedded with the hover bots. Figure 6.1 showcases the visualizations sketched, presenting different visual representations, embedding locations, colors and types of encoding. The extensive list and detailed descriptions of each visualization sketched can be found in Appendix A (Figure A.11, Figure A.12, Figure A.13). From this list, all uncommon or never used visualizations in real video games were excluded. Thus designs 10, 11, 19 and 20 were excluded. Visualization designs 2 and 5 embedded below the "feet" of the character were excluded since the robot is floating and the distance from the ground could be confusing for the players. Visualization 18 was also excluded due to the partial occlusion caused by the robot's design. Visualizations 13, 14, 16 and 17 were considered too small.

Among the designs sketched, I selected designs 1, 15 and 21 that have three different embedding locations. Figure 6.2 shows the chosen designs after a re-design. The first final design (Figure 6.2a) is a bar chart embedded *around the character*, specifically above its head. This design is the mainstream design for health representation and it was the original design used by the FPS Microgame. The second final design (Figure 6.2b) is instead *embedded in the character design*. The health percentage is embedded with the texture of

the robot that will change color according to health value itself. The higher the health value, the more the robot will be colored in red. Lastly, the third design (Figure 6.2c) is a donut chart *overlapping the hover bot*. This visualization rotates according to the robot to be always visible like the other two designs.

In order to reduce the number of influencing variables and design an experiment that was as controlled as possible, I used length encoding only to represent health. The visualizations also have red as unique color, with a black background for the visualizations not integrated in the character's design. Red was preferred over green since it is a color widely used to depict danger and enemies, so it suited the role of the robots well.



(a) Bar chart placed above the head (b) Colored texture embedded with the whole body of the character using length encoding. (c) Donut chart overlapping the body of the character using length encoding.

Figure 6.2: The three visualization designs about the robot's health level selected to study the impact of embedding locations on the visualizations' readability.

6.4. Choice of Percentages

In my study only 6 percentages values were tested: 18%, 32%, 43%, 58%, 72%, and 83% (see Figure 6.3). Those percentages are the same as the ones used by Yao et al. [66] in their evaluation of the readability of moving bar charts and donut charts. Since I am also evaluating bar charts' and donut charts' readability I decided to use the same values because they are fairly distributed in a 0% to 100% scale and match the design requirements of the RobotLife. Some other examples comes from the literature. Cleveland and McGill [10] used specific sets of 7 percentages for their position-length experiment: 17.8%, 26.1%, 38.3%, 46.4%, 56.2%, 68.2%, and 82.5%. Similarly, after observing that study participants tend to answer as factors of 5%, Kong et al. [31] used only four percentages: 32%, 48%, 58%, and 72%. The chosen percentages were equally distributed among all the enemies in the game scenario. To eliminate learning effect, each percentage value was assigned the same number of times to a randomly designated enemy. According to that, every robot had a different health value at each iteration of the game.

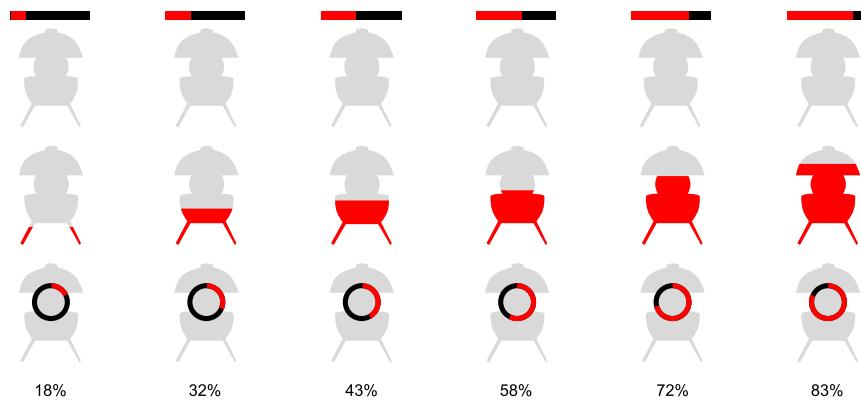


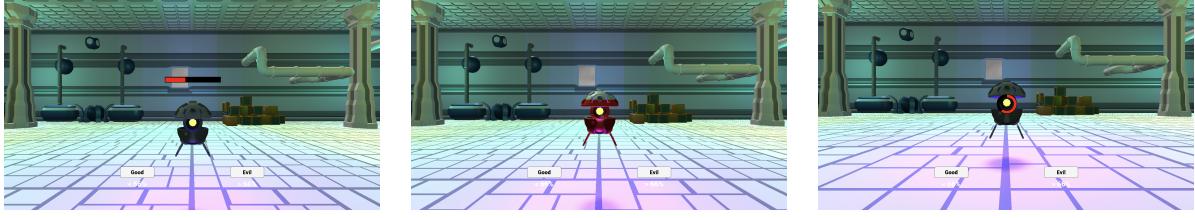
Figure 6.3: The three visualizations embedded in the robots representing the 6 percentages of health level used in the experiment: 18%, 32%, 43%, 58%, 72%, and 83%.

6.5. Training Session

After completing the tutorial and before playing the main game level with different health visualizations, the study participant had to complete a different training sessions for each visualization under evaluation. The training session was intended to teach the participants how to read a visualization and distinguish good robots from evil robots. A robot was presented on the center of the screen and its health level was visualized with one of the three visualizations, according to the current visualization under test (see Figure 6.4). Each training session was about a specific embedding location, so only one visualization design at the time was embedded in the robot. Two buttons "Good" or "Evil" allowed the player to select the correct choice according to the value represented by the visualization. The result was displayed right after the selection, together with the actual health percentage represented. The health level was randomly generated from a range between 0% and 100%. After 5 correct answers, the participants were allowed to quit the training or keep going until they felt confident.

6.6. Gameplay Flow

The core of the experiment was playing RobotLife. The game was implemented to be self explanatory so that the participants could play without external intervention or help. After launching RobotLife, the game asks for a participant ID to initialize the settings of the experiment. Since I wanted to test 3 different visualizations inside the game, I assigned to each visualization a condition: Condition A represents the bar chart embedded around the character, Condition B represents the integrated texture visualization and Condition C



(a) Screenshot from the training sessions of the visualization embedded *around the character*. (b) Screenshot from the training sessions of the visualization *lapping the character*. (c) Screenshot from the training sessions of the visualization embedded *integrated in the character design*.

Figure 6.4: Screenshot of the training sessions inside RobotLife for the three visualization under test.

represents the donut chart overlapping the character. The order in which the 3 conditions were presented to the participants depended on the participant ID. ID 1 was associated to the "ABC" order, participant ID 2 was associated to the "ACB" order, participant ID 3 was associated to the "BAC" order and so on, following a classic Latin Square algorithm. Orders and consequent scene flow management were calculated at run time. Once the game was initialized, general instructions were provided to the players. After, participants were required to complete the Tutorial at least one time. They were free to repeat the tutorial as many times as they want, until they felt confident. Once the tutorial was completed, RobotLife presented three different blocks, once per each condition (A, B or C). Each block comprised a training session, a 5 minute gameplay of the main level and an in-game questionnaire. The training session was introduced by a full explanation of each visualization, showing the embedded visualization with hover bot representing values from 0% to 100%. The maximum number of training trials was not defined and each participant could decide whenever he/she was comfortable with the visualization enough to stop training. The gameplay consisted in playing RobotLife for a duration of 5 minutes per block. Participants were free to play the game as many times as possible inside the 5 minute time range, at their own pace according to their capabilities. Specifically, if a player died or won before 5 minutes, he or she had to keep playing the same level until the deadline of 5 minutes. At the timeout, a in-game questionnaire was presented directly inside RobotLife. After submitting the answers to the in-game questionnaire, a new block was proposed. After all the three blocks were completed, a message notified the end of the gameplay of RobotLife.

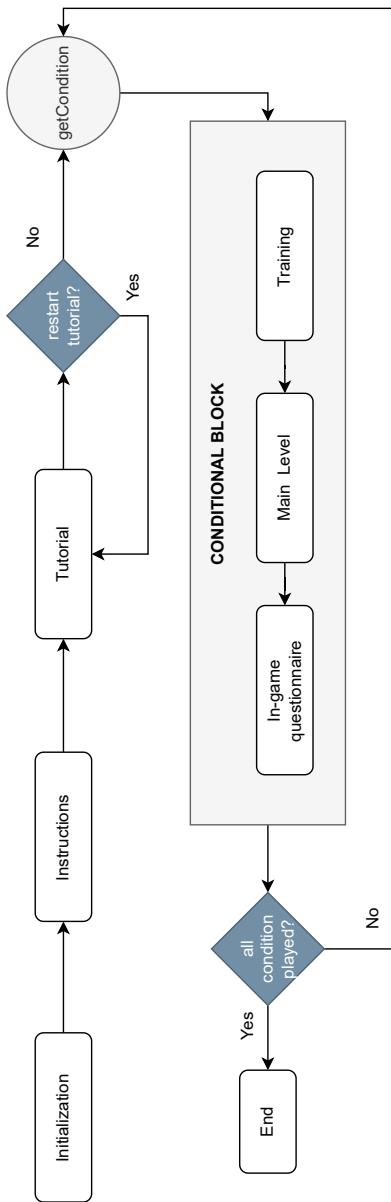


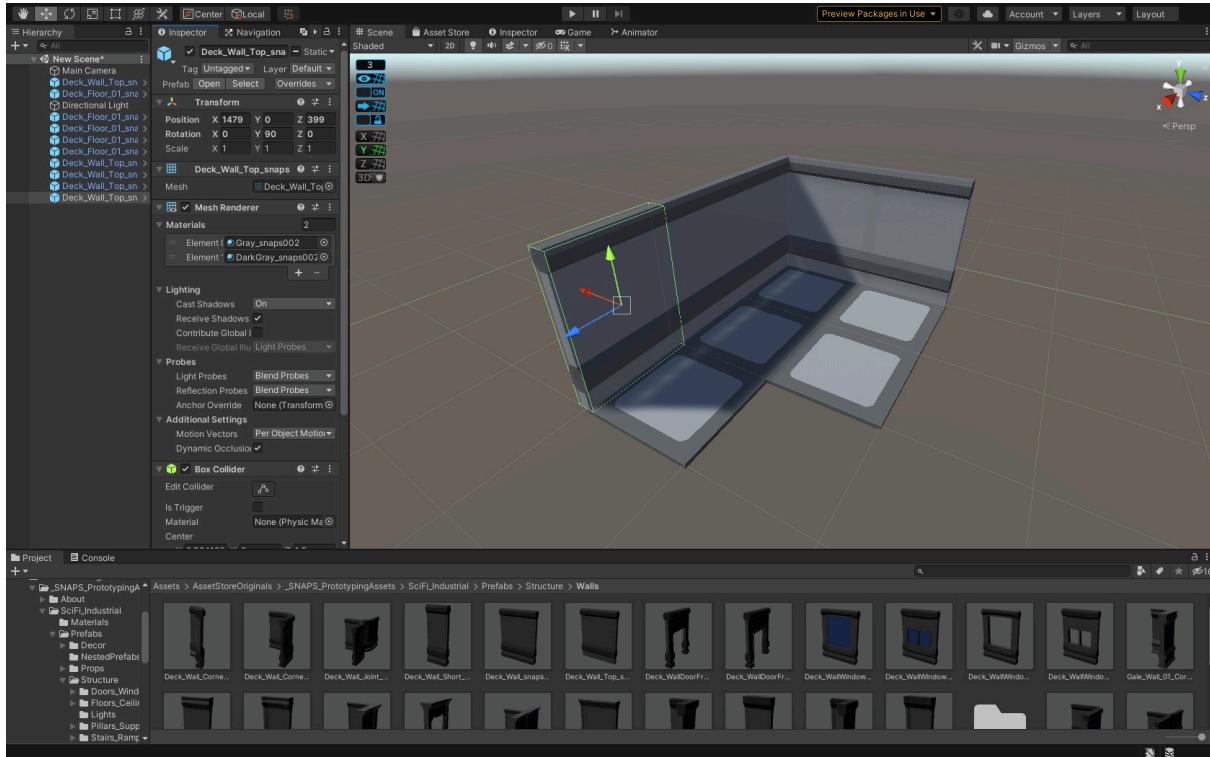
Figure 6.5: Diagram of the gameplay flow of RobotLife. After the initialization, instruction and tutorial, the game comprised three different blocks, one per each visualization under test. Each block comprised a training, a 5 minutes gameplay of the main level and an in-game questionnaire.

7 | RobotLife Implementation

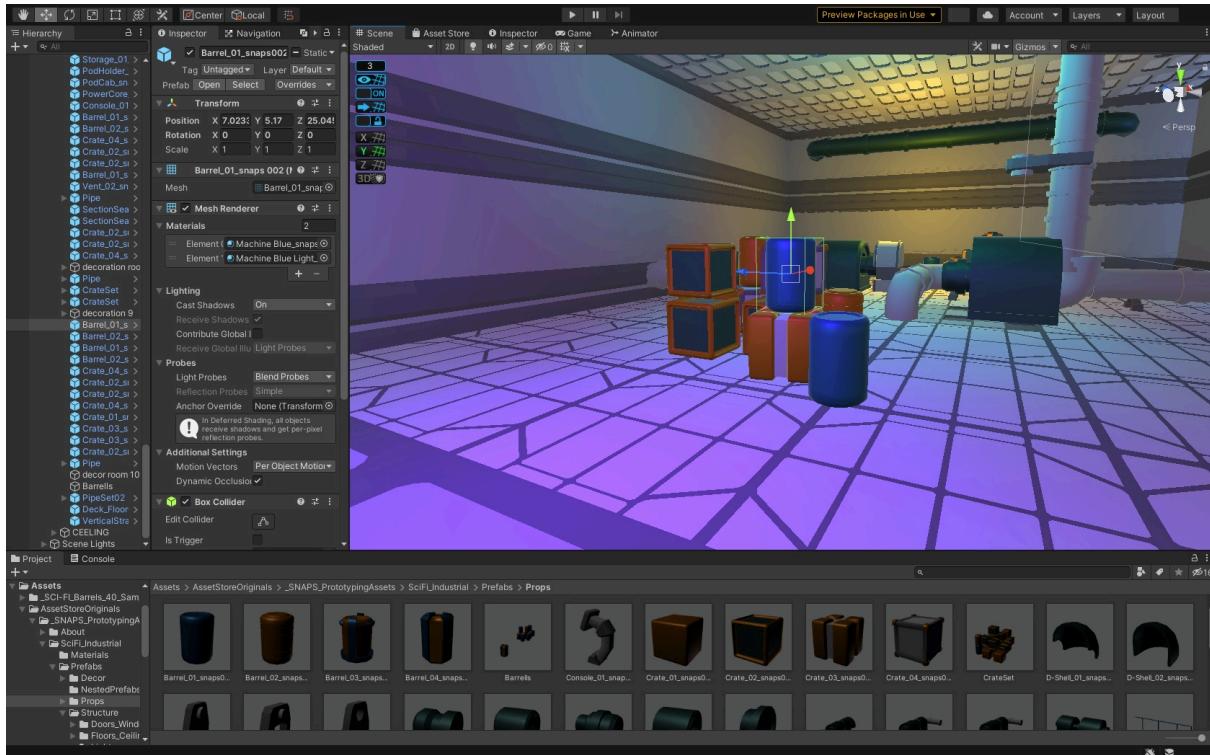
RobotLife was implemented using Unity version 2020.3.31f1. The game was deployed for both Mac OS and Windows and it uses mouse and keyboard as input devices. The game can be played with both *azerty* and *qwerty* keyboard layouts. The programming language used is C#, the native language of Unity. RobotLife was built starting from the open-source FPS Microgame Unity Learn template [48], that allowed me to start the implementation with the basic game mechanics already in place. In the following sections I describe some implementation details such as the game world implementation, the game mechanics implementation and the visualization implementation. Moreover, I report here the implementation of a second level of RobotLife that was not used for the scope of this thesis but that will be used in the future to conduct further exploration of situated visualization in motion for video games.

7.1. Game World Implementation

Art production can take up to the 50% of the time devoted to game development. To create the environment of RobotLife, that is set inside a factory, I used an asset bundle called *Snap Prototype / Sci-Fi /industrial* made by Unity Asset Store Originals [41]. The advantage of this Snap asset bundle is the modularity of its game-ready assets that include environment assets, props and more. Span assets are compatible with ProBuilder [52] that is a Unity level design tool, optimized for building simple geometries and build quick prototype structures. The Snap Prototype | Sci-Fi /industrial bundle offered assets like floor tiles, ceiling tiles, walls and props (i.e. barrels, tubes and more) easy to combine thanks to the snap grid mechanism of ProBuilder. This Unity feature allowed me to reduce the art production time and create a detailed level environment. Figure 7.1a shows the game world design at a very early stage. The figure shows floor tiles and wall placement inside the default scene template of Unity. Figure 7.1b shows a later stage, while placing props such as barrels and boxes around the rooms created. This figure shows also the use of lights inside the environment. Specifically, I used only point lights of different colors and intensity, to enhance the Sci-Fi ambience and create high color contrast in the scene.



(a) Game world implementation at the early stage, showing floor tiles and walls placement.



(b) Game world implementation at a later stage, showing props placement and the use of lights.

Figure 7.1: Screenshot from Unity, showing the game world implementation in RobotLife.

7.2. Game Mechanics Implementation

The implementation of RobotLife started from the open-source code of *FPS Microgame* developed by Unity Learn. To adapt the microgame mechanics to the study requirements I developed and modified several scripts in C#. GameObjects are the fundamental objects in the Unity scenes [51]. The behavior of each GameObject is defined by the components attached to it. It is possible to assign custom components by adding scripts to any GameObject. The default template of a script presents a `Start()` method that is used for the initialization of the GameObject, and an `Update()` method that is called once per frame.

Here I present the most relevant scripts that I developed: `ObjectiveKillEvilRobots.cs`, `ObjectiveSaveGoodRobots.cs`, `EnemyMobile.cs` and `EnemyController.cs`. The full code of the scripts can be found in section A.5.

7.2.1. ObjectiveKillEvilRobots.cs

This script defines the behaviour of the primary objective of the game and it is responsible for managing the game status and updating the primary objective UI notification. The `assignRobotsAffiliation()` method takes care of the initialization of the robots affiliation. Receiving the `GameObjects[] robots` array, this method shuffles their order in the array and assigns a starting health value to each robots, changing the robots affiliation accordingly. To have a fair distribution of values, I used only 6 predefined health values in a range from 0% to 100%: 18%, 32%, 43%, 58%, 72%, 83%, as justified in section 6.4. In the main game scene, 24 robots are available, thus each health value is assigned to 4 robots. Therefore there are 8 evil robots in the scenario and 16 good robots. The `ObjectiveKillEvilRobots.cs` script adds an event listener for `EnemyKillEvent` events. The method `OnKill(EnemyKillEvent evt)` reacts to the aforementioned event type and verifies if the killed robot is good or evil to update the game status and eventually declare the completeness of the primary objective, leading the player to victory.

7.2.2. ObjectiveSaveGoodRobots.cs

This script defines the behaviour of the secondary objective of the game and it is responsible for declaring game over if the player hits more than 2 good robots. The `Start()` method inside the script adds an event listener for `DamageEvent` events that is produced every time the player hits a robot. The `OnDamage(DamageEvent evt)` method reacts to this event type and checks the affiliation of the damaged robot. If the robot is good, a counter increases the number of good robots damaged and if the number its equal to the number

of mistakes allowed (in the implemented level, this number is equal to 3), it declares the `Health.Kill()` method to kill the player. In this case, the player dies and its game over.

7.2.3. EnemyController.cs

This script was already implemented in the FPS Microgame template. Here I describe the modification that I introduced to the script that resulted in some of the core game mechanics of RobotLife. The hover bot GameObject has a NavMesh Agent component attached that allows it to navigate inside the game world thanks to a baked NavMesh. A NavMesh is a data structure that represents the walkable surface of the level geometry and allows NavMesh Agents to find path from one walkable location to another [49]. The `EnemyController.cs` script prior to modifications, made the robots move towards the player GameObject whenever it was at range, starting to shoot at it immediately. I modified this behaviour inserting a boolean `attacksOnlyOnDamage` that allows the robots to react only when the player attacks it first. This boolean allows to call the `DetectionModule.HandleTargetDetection()` either if the boolean is set to false, inside the `Update()` method or, only on a `DamageEvent` event, handled in the `OnDamaged()` method. This modification allows any robots to attack the player only when damaged first, giving the player more freedom to move around the game world and read the visualization under evaluation. In this script I implemented another important method that is called by the `EnemyMobile.cs` script: the `RandomNavDestination(float radius)` method. This method returns a random destination inside the NavMesh of the level geometry, inside a range defined by the `float radius` parameter. This method is important because it contributes to the ability of the robots to move freely in the game environment, adding autonomous motion factors to the game.

7.2.4. EnemyMobile.cs

This script was already implemented in the FPS Microgame template. Here I describe the modification that I introduced to the script to allow the robots to navigate freely inside the game world. This modification allows to generate autonomous motion factors that influence the visualizations readability. This script assigns an `AISystem` to the robots. There are three states possible: `Patrol`, `Follow`, `Attack`. Robot's free navigation inside the NavMesh takes place in the `Patrol` AISystem. This script allows to set a `navigationRadius` from the Unity inspector, to define the distance allowed to be navigated in the `totalNavigationTime`, also modifiable from the inspector. The `UpdateCurrentAISystem()` method, defines the navigation behaviour in the `Patrol` state, calling the `EnemyController.SetNavDestination()` method and passing as parameter

the random destination returned by the `EnemyController.RandomNavDestination(radius)` method. In this way, robots update their destination every `totalNavigationTime` seconds and move autonomously around the environment.

7.3. Visualizations Implementation

Here I present the implementation of the three visualizations under evaluation. All the visualizations encode the health of the robots using only length. Moreover, red was used as the unique color to ensure a high contrast with the developed game world palette.

Figure 7.2a shows the bar chart visualization embedded around the character implemented. This visualization is implemented using a `Canvas` with `Render Mode:World Space`, meaning that the canvas is positioned in the world space and not overlapping the main camera of the scene. Two images are attached to this canvas: a background image in black, and a fill image in red. The background image defines the full length of the bar and provides a black border around the visualization. The fill image in red defines the actual health level and it is defined with `Image Type:Filled` to be able to modify the `Fill Amount` by code when the robot is damaged. The script `WorldspaceHealthBar.cs` takes care of modifying the fill amount and rotates the health bar to always face the camera/player. This result is achieved with the use of the `Transform.LookAt(Camera camera)` method.

The colored texture visualization integrated in the character design is instead implemented using a `shader` to modify the material color of the hover bots as shown in Figure 7.2b. The hover bot prefab comprises three parts: top, eye-ball and bottom. The implemented shader is applied to each material of those robots' parts. The shader allows to set a default color, a highlight color and an highlight percentage. By modifying the highlight percentage, the highlight color area in the texture increases or decreases accordingly. The shader identifies local min and local max points and performs a calculation to identify the point where the highlight percentage should be in the 3D world. Local min and local max differs from top, eye ball and bottom and it is possible to set them directly from the Unity inspector. The full shader code is available in the Appendix A - section A.6. Looking at Figure 7.2 it is possible to notice that the red color applied to this visualization appears slightly darker than in the other two visualizations. The color code used is the same for each of the visualizations (RGB 255, 0,0) but due to the rendering conditions and the texture map applied to robot itself, by integrating the red directly on top of the texture map, the color is subject to the texture shading map.

Lastly, the donut chart visualization overlapping the character is presented in Figure 7.2c. This visualization is implemented using a `Canvas` as Visualization A. In this case, the background and fill images have a circular shape and the fill amount is calculated with a



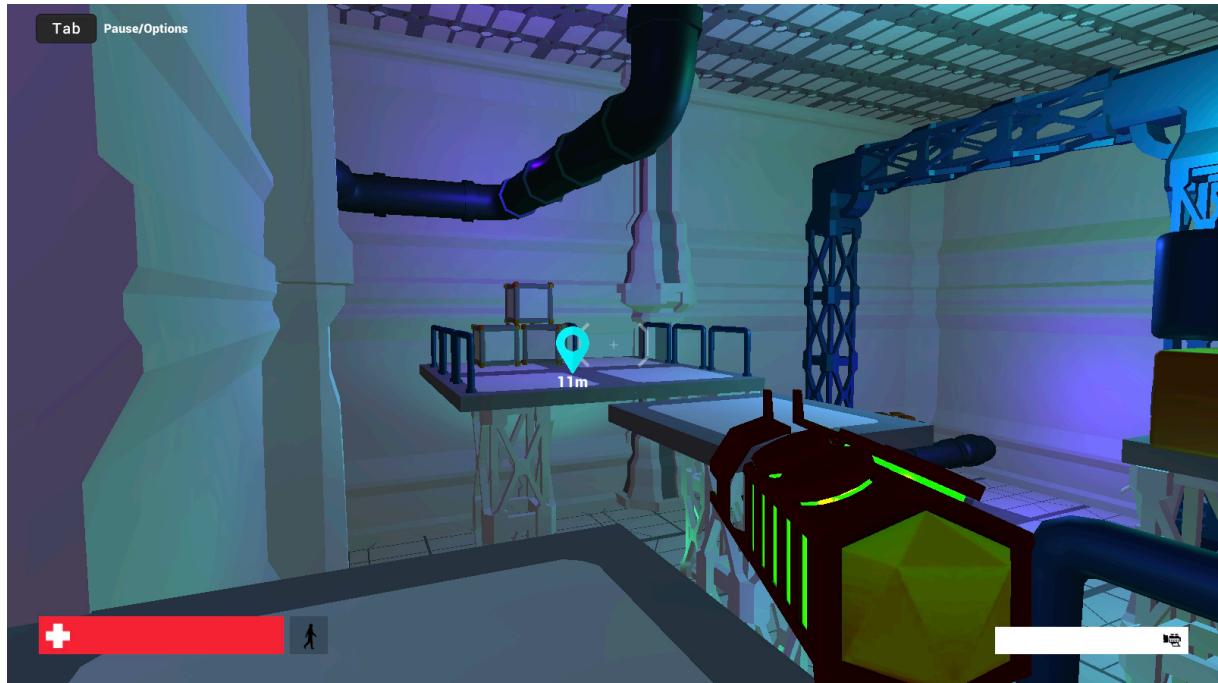
(a) The implemented bar chart visualization embedded around the character. (b) The implemented colored texture sign. (c) The implemented donut chart overlapping the character.

Figure 7.2: Implemented visualization designs of the robot’s health for the three conditions under evaluation. Each visualization in this figure shows an health value of 66%. Those figures are produced with Unity and show the actual GameObjects inside RobotLife’s levels.

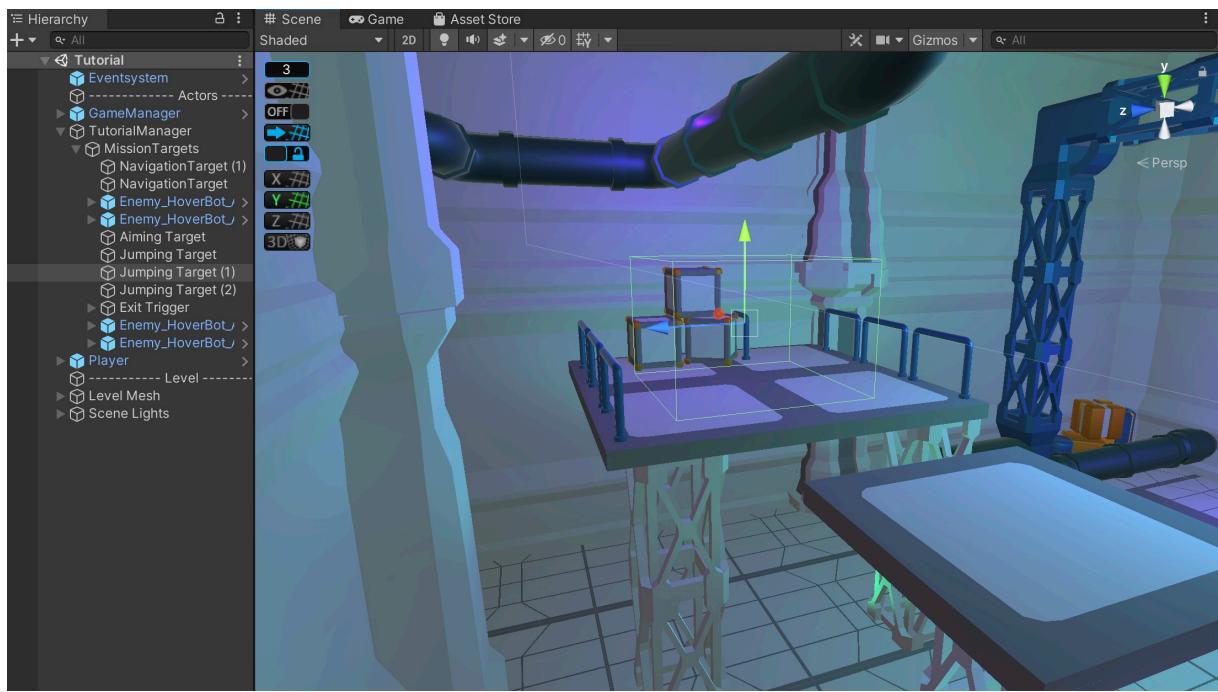
radial function over 360 degrees. Moreover, only the `Transform.LookAt(Camera camera)` method was not sufficient in this case because the visualization could be occluded by the robot body in case of rotation. To solve this problem, the `WorldspaceHealthBar.cs` when the `isOverlapping` boolean is true, calculates the relative position of the camera/player with respect to the visualization’s transform and calculates a Quaternion expressing the rotation needed. This rotation is then applied to the `Canvas` containing the visualization images in order for the donut chart to always face the camera/player.

7.4. Tutorial Implementation

Following the design choices discussed in section 5.6, I implemented the tutorial as a fully playable level. The tutorial scene shares some core GameObjects with the main level such as a `GameManager`, a `Player`, an `Objective`, an `EventSystem` and a baked NavMesh. Moreover, the main elements of the tutorial are the tutorial triggers, the waypoint indicator and the instruction panels. A tutorial manager is represented by the `ObjectiveTutorial.cs` script that implements the abstract class `Objective.cs`. This script takes a list of waypoints (tutorial triggers), a waypoint indicator for the UI, a list of instruction panels and the total number of triggers to consider in the scene. Therefore, it is possible to add or remove instructions and waypoints at any time through the inspector. The first instruction is displayed directly by the `Start()` method. It assigns the first objective to the waypoint indicator to track the object in space, displaying the distance between the player and



(a) Example of waypoint indicator in the tutorial scene UI.



(b) Scene tab showing an example of Box collider used for the tutorial triggers. Inspector tab showing the structure of the Tutorial scene.

Figure 7.3: Screenshot from Unity, showing the implementation details of the Tutorial scene of RobotLife.

the object itself. The `MissionWaypoint.cs` is taking care of updating the waypoint UI indicator, making sure it remains always on screen even if the player is not facing the objective directly (see Figure 7.3a). This result is achieved by clamping the indicator's x and y position attributes to the actual min and max values of the screen size. The `ObjectiveTutorial.cs` script exposes a method called `displaNewInstruction()` that is called by the tutorial triggers when needed. The `TutorialTrigger.cs` script is attached to each trigger GameObject and with the use of Box Colliders, it exploits the library function `OnTriggerEnter()` to detect when the player enters the trigger. On enter, the triggers call the `ObjectiveTutorial.displaNewInstruction()` to display a new instruction panel if needed and update the current objective. The same behavior is produced by the `TutorialEnemy.cs` script that calls the `ObjectiveTutorial.displaNewInstruction()` when the enemies in the tutorial die.

7.5. Second Level: Game element types visualizations

With the aim of reporting all the features already implemented inside the RobotLife project, I present in this section a second game level that was not used in the study designed for the purpose of this thesis. This second level was intended to evaluate the impact of different *visual representations* on the readability of situated visualizations in motion for different game element types (resources). Particularly, this level can be used in future work to complement the results of the study designed and to further explore the impact of different factors on visualizations' readability.

In this level, evil robots managed to break the main engine of the factory. The player has to repair the engine by picking up some resources and bringing them back the engine room to fix the engine itself. The level proposes three types of resources, namely batteries, gears and bolts. The player has to navigate the scenario, kill the robots attacking him/her and pick up the correct resource types needed to fix the engine. The level mechanics in this case focus on the visualizations of the resource types, that the player has to correctly distinguish. In fact, the player has a limited number of pick up actions and a legend placed on the game UI shows the resources needed to fix the engine. In this way, players have to correctly identify each resource type before making their decision to pick up an object because otherwise he/she will run out of pick up actions before being able to fix the engine with the correct resources.

Since this level will serve to explore the impact of different visual representations, I designed three visualizations embedded around the resources. All the resources are represented by an anonymous crate object. The first visualization is a simple label with the name of the

resource type displayed (Figure 7.4a). The second visualization was a colored circle above the object where each type was represented by a different color (Figure 7.4b). The third visualization was a sign representing the type of resource by different shapes (Figure 7.4c).

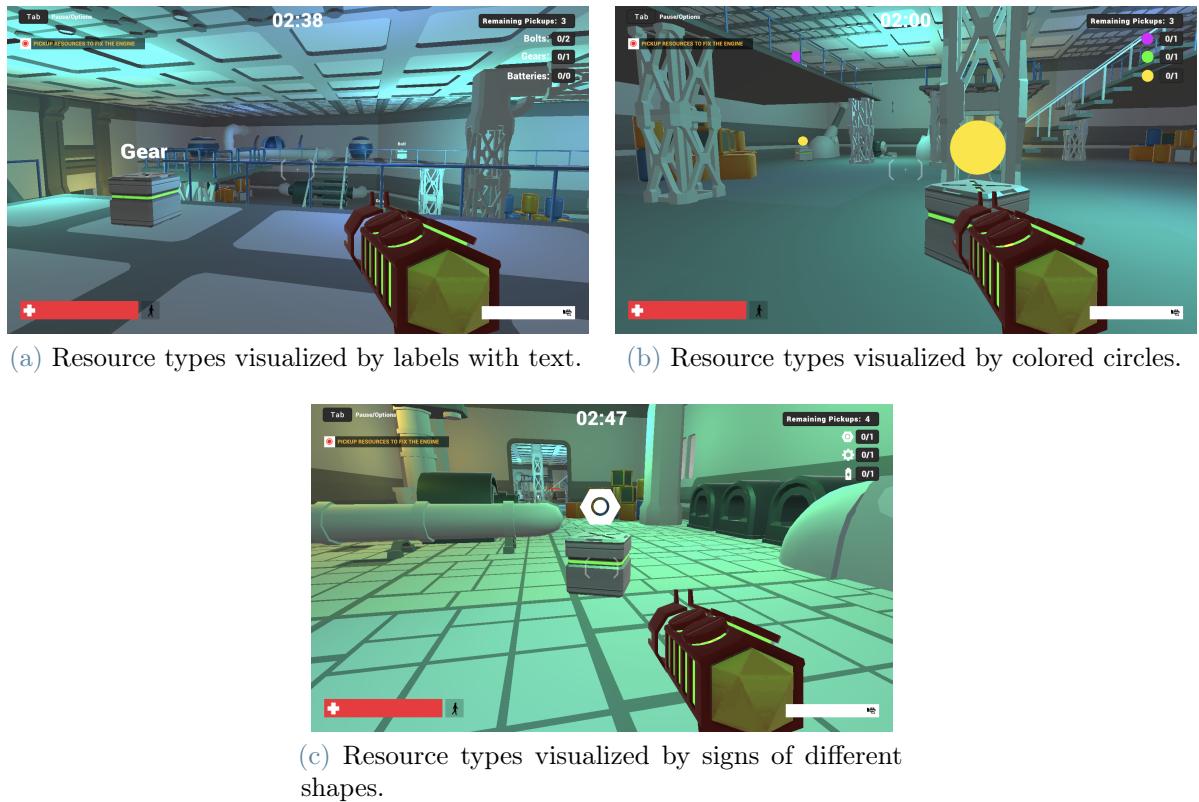


Figure 7.4: Second Level of RobotLife developed. In this level the player has to pick up some resources to fix the main engine of the factory. The resources are of three types and the type is visualized with three different visualizations to test the impact of different visual representations' readability.

A new game map was implemented for this game level. With respect to the level focusing on the visualizations of the health level of the hover bots, this level is smaller but still big enough to propose a challenging navigation inside the 3D environment. The game environment is built using the same asset bundle *Snap Prototype / Sci-Fi /industrial*, as well as the ambient colors (lighting).

The `ObjectiveResourcesPickUp.cs` script is mainly responsible for the game mechanics implemented in this level (script available in the Appendix A - section A.5). This script takes care of three things:

- **Creating the list of resources to pick up:** the `createPickupList()` method creates a randomly generated list of resources that the player has to pick up in order

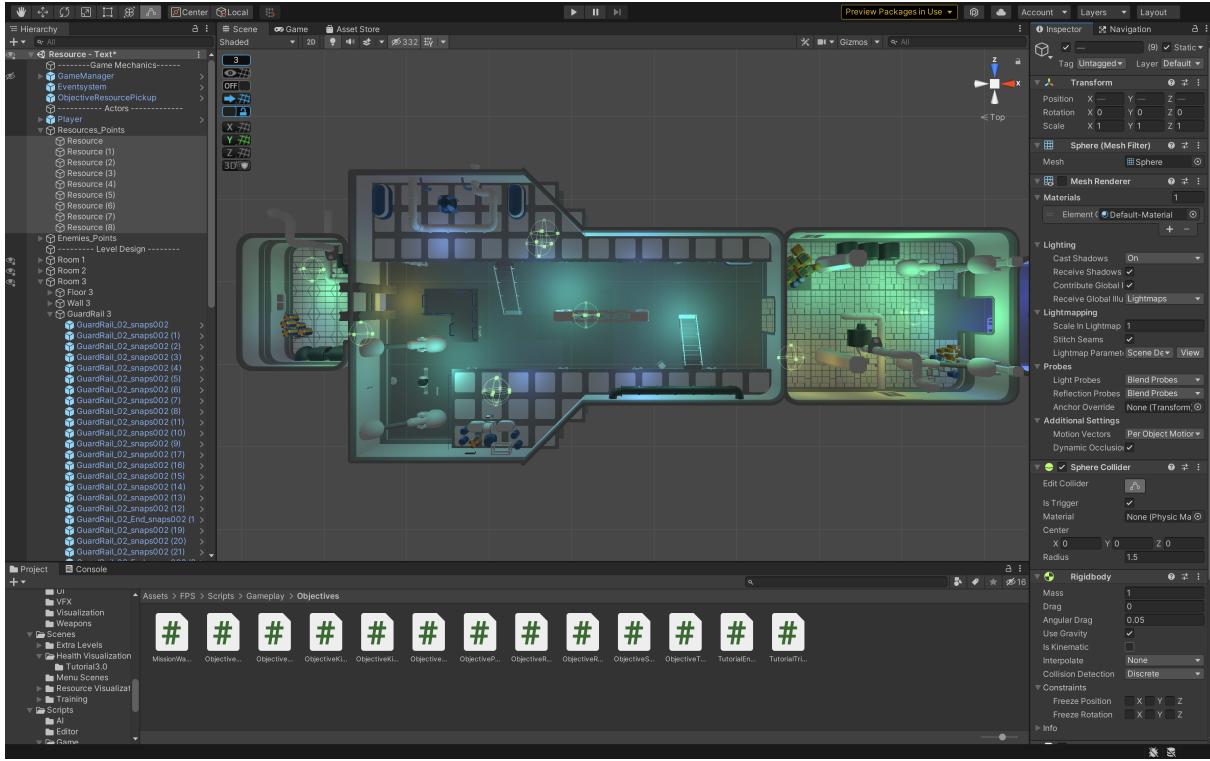


Figure 7.5: Unity editor with the second level scene opened, showing the level map with the resource spawn point highlighted by the mesh colliders.

to complete the objective of the level and win. The list is created by generating an `int` from 1 to 3 where each number corresponds to a resource type (1 = bolt, 2= gear, 3 = Battery). To handle the resource types I implemented an `Enum` class containing the definitions for Bolt, Gear and Battery. The script holds 3 counters to save the number of objects per resource type generated by the random extraction.

- **Display and create the resources:** After the list of resources to pick up is generated, the `displayResources()` method takes care of instantiating the actual prefabs of the resources inside the game environment. In the scene, I instantiated some empty gameObjects that acts as parent objects (see Figure 7.5). The scripts instantiates the different resources types prefab according to the visualization used in the level to display the resource type. In this way, the resources are placed randomly and no learning effect is possible.
- **Handle pick up events:** the `OnPickupEvent(PickupResourceEvent evt)` method reacts to the player picking up some resources. this method controls if the player completed the objective of the game by counting the resources remaining to pick up, checking per type of resources needed. If the player has enough pick up actions left to complete the action, this method destroys the object prefab and updates the

count of resources collected. If the player has no more pick up actions, the method throws a new `PlayerDeathEvent` to declare game over.

Moreover, the hover bots in this level are present and their health starts always from 100% and it is represented with the bar chart visualization embedded above the head of the characters. The robots are spawned cyclically by the `EnemyGenerationPoint.cs` script, from four spawn points in the scenario so that every time a robot is killed by the player, a new robot will appear after a generation interval, set to 15 seconds in the inspector. In this way, players have to deal with shooting at the robots and survive their attack, maintaining the experience of a real FPS game even though the main objective focuses on picking up the resources.

8 | Pilot Study

After designing a study to evaluate the impact of embedding locations on the visualizations' readability for situated visualizations in motion in the context of games, we¹ conducted a first initial pilot and then asked the ethics committee for approval in order to conduct a full study. Unfortunately, we did not receive the approval in time, therefore we decided to proceed by running a second extensive pilot due to the limited time span remaining to conclude my thesis. This extensive pilot study took the form of a real experiment. In this section I describe the consideration emerged from the pre-pilot study and report the main pilot by describing the pilot participants, the experimental software and apparatus, the study procedure and the analysis approach.

8.1. Pre-Pilot

A pre-pilot study was conducted in order to test and evaluate the study design. The pre-pilot was centered around the gameplay of RobotLife. The main goal was to assess if the game instructions were clear, understandable and self-explanatory. Also, we tested the usability of the game commands, the map design and visualizations quality and implementation. The first pilot was conducted in Bordeaux and we recruited 5 participants with a data visualization background. Unfortunately, not all the participants were familiar with video games so some of them experienced issues due to their unfamiliarity with the domain and the game practices. Participants were asked to read carefully all the instructions inside the video game and provide feedback using think out loud technique. Then, we asked participants to go through the three conditional blocks of RobotLife, providing feedback on both the visualizations and the overall game experience.

The most important issue that emerged was that the game resulted very computationally heavy and so not all the participants' personal computers were able to run RobotLife at a decent speed. Analyzing the software performances, the problem was resolved by

¹The study was run under the supervision of my advisors. Any use of we in this chapter refers to: Federica Bucchieri, Lijie Yao and Petra Isenberg. I was solely responsible for the experiment software implementation, study design and result analysis.

reducing the amount of lights in the scene and removing the ability of the lights to cast shadows in the environment. Also, the overall number of robots was reduced from an initial number of 30 to a final of 24. With those changes, the speed of the game increased significantly. Overall the instructions seemed pretty clear except from the tutorial. Initially the implementation of the tutorial comprised some circular areas that the players had to walk in to show a new instruction on screen. Those areas were replaced with the waypoint indicator that thanks to the distance counter from the player, results intuitive and more easy to understand. Also the quality of the instructions panel was improved both in term of graphic and in quality of the instruction texts. Regarding the training sessions, it was not initially outstanding that after 5 correct answers the participants were free to quit the training and more on. Therefore, a new panel appearing almost full screen was implemented, preventing any other action other than reading the message explaining the possibility to train more or quit the training session. Also, 2 out of 5 participants experienced some navigation issues due to the complexity of the level map. Therefore the map was re-designed and simplified to be accessible also to novice players. Regarding the hover bots, it was not initially clear if by reducing the health level below the threshold of 66% they will automatically turn good. The rule that evil robots could not turn into good robots was then made clear while explaining the game scenario. Last, the participants experienced some issues related to the visualizations design. The donut chart overlapping the body of the robot was subject to some distortion issues due to the camera orientation. This problem was solved by correcting the script taking care of the visualization orientation with respect to game camera. Also, regarding the visualization integrated in the hover bot design, due to the bot animation, it was not always possible to see the eye-ball of the character. Therefore I modified the rigged skeleton of the hover bot 3D model to make the visualization more easy to read also when around the eye-ball level.

8.2. Participants

For the extensive pilot study I recruited 12 participants. Participants could only take part in the experiment if they had no sort of problem with moving images on screens, if they had no motion sensitivity problems (vestibular disorder) or photosensitive epilepsy and had a normal or corrected-to-normal vision. Also participants were of legal age (above 18) and able to speak English. Participants with previous gaming experience with any kind of video games, not only FPS games, were preferred. Participation was voluntary and the study was anonymous, no personal data has been asked or recorded. I recruited 5 female and 7 male participants, with an average age of 25 years (see Appendix A - Figure A.17). Participants were not very familiar with video games playing, since they self reported

their level of ability in playing games on a scale from 1 to 7 and on average the level was 3.80. Their frequency of playing video games was on average once or twice every three months. Instead, their familiarity in playing with FPS games specifically was also low (3.30/7) and their frequency of playing those kind of games was once or twice in a year. The participants recruited played mostly Role Playing Games (RPGs), Platformers and Action/Adventure video games.

8.3. Experimental Software and Apparatus

The pilot study was conducted in a laboratory setting. Each participant conducted the experiment on a MacBook Pro 16-inch (2019) with a 2,6 GHz Intel Core i7 6 core processor. Participants were asked to complete some questionnaires on Google Forms and then play RobotLife with keyboard+mouse input. An external mouse was preferred to the integrated MacBook Pro track pad for ease of use while gaming. In this case, the version of RobotLife used was the one deployed for MacOS and qwerty keyboard layout support. Moreover, the game was set to a 16:9 screen resolution and it was played with maximum screen brightness.

8.4. Study Procedure

First, participants had to agree to a consent form. Second, they completed a pre-questionnaire about their demographic data - age and gender - and gaming habits like frequency of playing games, frequency of playing FPS games and more (see Appendix A - section A.7). This pre-questionnaire was intended to provide an overview of the participant experience in playing video games and specifically in playing FPS games with keyboard+mouse input on PC. To objectively evaluate the player experience I considered the frequency of gaming declared by the participants (i.e. daily, weekly, once a month). After completing the pre-questionnaire, RobotLife was launched and the software guided the players through the experiment under my constant supervision. While playing, a logging system recorded all the in-game events producing a log file per participant. On average each participant played RobotLife for 27 minutes, considering the initial tutorial, the three training sessions and the time needed to read the various in-game explanations of each part of the experiment. To complete the tutorial the participants needed on average 2 minutes and 35 seconds. After the tutorial, the three RobotLife's blocks were evaluated independently. At the end of each conditional block, participants had to complete an in-game questionnaire. The in-game questionnaire comprised 2 likert scales with 4 and 5 items each (see Appendix A - section A.8). After completing the gameplay of RobotLife,

participants completed a post-questionnaire asking them to rank the three visualizations according to different aspects (see Appendix A - section A.9). Lastly, participants took part in an interview to gather qualitative data, feedback and insights about their strategies while playing RobotLife and general comments about the visualizations.

8.5. Analysis Approach

I collected both quantitative and qualitative data. First, to provide an overview on how the participants performed while playing RobotLife, I implemented a logging system that recorded every salient action occurred during the gameplay. Particularly the logging system implemented is based on an event management mechanism: at every event broadcasted by the centralized `Event Manager` class, it stores a new record saving the type of event occurred, its timestamp and the visualization under test. Examples of the data collected are the number of evil robots killed per level, the number of good robots damaged, the number of winnings and more. Log files were then processed using a Python script to extract aggregated data and then analyzed to generate statistics about the participants' performances in the experiment.

Second I gathered ordinal data from the in-game questionnaire at the end of each block of RobotLife. The in-game questionnaire comprised two Likert scales of 4 and 5 likert-type items respectively. The first likert scale evaluated the visualizations' readability and the second one the visualization's aesthetics. For the aesthetic Likert scale I used the validated BeauVis rating scale proposed by He et al. [22] to assess the aesthetic pleasure of a visual data representation.

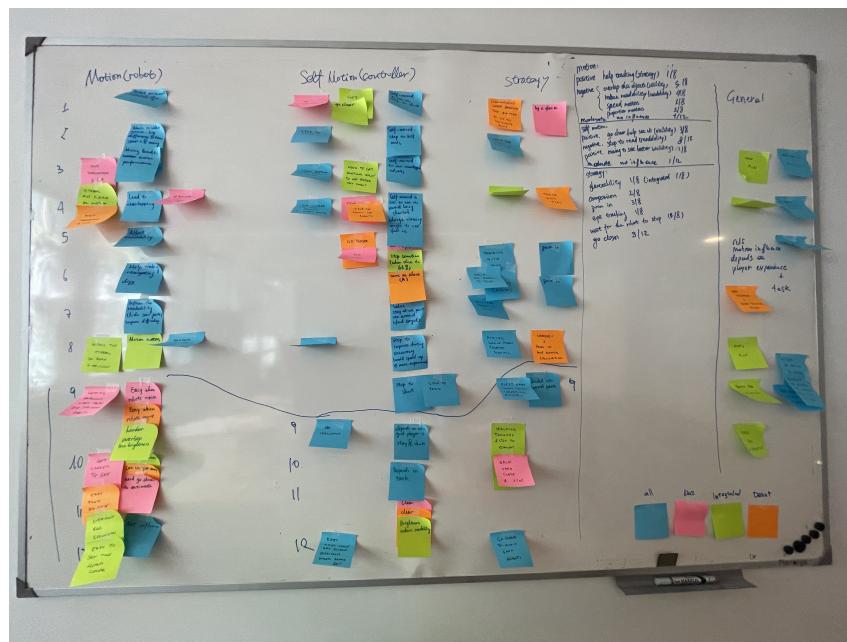
Furthermore, I collected ordered data from the post-questionnaire. Participants were asked to rank the three visualizations under evaluation in terms of readability, aesthetic, support towards achieving the game goal and for the overall game experience.

Last, I gathered qualitative data by interviewing participants at the end of the experiment. Particularly, I asked about the influence of motion factors, the impact of the different embedding locations and the influence of other contextual factors on the visualizations' readability. Examples of other factors were the game mechanics, the player's experience or the visualizations occlusion due to other game objects. The data gathered during the interviews was subsequently thematically coded by the examiners and divided in six categories. Those categories derived directly from the topics of the questions asked. Particularly, those categories were my main interest while evaluating the different visualizations proposed. My goal was to understand the overall perceived readability, how well those visualizations matched the aesthetic of the game and how much motion factors (both depending on the robots' movements and on the player's control) impacted the visualizations' readability.

Practically, each participant's statement was recorded and divided first per visualization with a specific post-it color and then attached to a whiteboard in one of the six categories listed on the board. Pink was used to represent the bar chart visualization embedded *around the character*, green represented the colored texture visualization *integrated in the character design* and lastly, orange was used for the donut chart visualization *overlapping the character's body* (see 8.1a). General comments not associated to a specific visualization were recorded in blue. Each category was then analyzed separately to point out subcategories. For example, inside the readability category, a few comments were about the shape of the representation and how it influenced the visualizations' readability so I grouped the participant's answers to code the pro and cons of each visualizations (see 8.1b).



(a) First three categories of the interviews' thematic coding.



(b) Last three categories of the interviews' thematic coding.

Figure 8.1: Thematic coding of the interview notes. Each visualization under evaluation was represented by a specific post-it color. The notes were categorized according to 6 categories: Readability, Aesthetics, Embedding Locations, Robot's Movement, Self-Movement, Strategy.

9 | Results

In this section I present the results of the pilot study conducted to evaluate the impact of different embedding locations on three visualizations' readability while in motion in the context of video games. First, I present the result from the quantitative data analysis to then complement those results with the insight gathered by interviewing the study participants.

9.1. Game performances

Analyzing the log files recorded by the gameplay of RobotLife, those are the results emerged (see Appendix A - Figure A.16). On average, with the bar chart visualization embedded around the character, participants replied correctly 7.60 times during the training session and only two participants made a mistake during this phase. During the main game level, participants killed 16 evil robots and damaged 2 good robots. On average the level was won 1 time and only 2 participants lost one time each.

Using the colored texture integrated in the character's design, participants answered correctly to 8.25 trainings and only two participants made one or two mistakes during this phase. Moreover, 15 evil robots have been killed and only 1 good robot damaged. On average the level was won one time and only 3 participants lost one time each.

Last, using the donut chart overlapping the character, during the training session participants replied correctly 6.60 times and only two participants made one or two mistakes. During the game play 18 evil robots have been killed and around 3 good robots were damaged. Also, players won on average 1.40 times and 4 players lost one or two times.

As suggested by those results, the visualization overlapping the character - the donut chart - was the visualization that generated more game events, resulting in a very dynamic gameplay. With this visualization players played the level on average 3 times in five minutes. The maximum of evil robots killed was 25 using the visualization overlapping the character, with respect to 23 using the bar chart embedded around the character and 21 using the colored texture integrated in the character's design. Regarding the number of

good robots damaged instead, the maximum was 7 using the donut chart, followed by 4 using the bar chart and 3 with the colored texture.

9.2. In-game questionnaire results

The in-game questionnaire was presented to each participant at the end of each conditional block inside RobotLife. Consequently, the participants completed the questionnaire for each visualization in a different order, according to their participant ID assigned with a Latin Square algorithm. This means that in some cases, a visualization was evaluated without having seen the other ones yet. This questionnaire was composed by two likert scales of 4 and 5 likert-type items each and referred to each specific visualization individually. The answers collected during the in-game questionnaire are available in the Appendix A - Figure A.18.

9.2.1. Readability

The readability evaluation contains 4 likert-type items including readable, clear, understandable and intuitive (see Appendix A - Figure A.14). The vote distribution for each item is shown in figure Figure 9.1. Following, I discuss how participants reported per item.

Readable: Participants reported the bar chart with an around embedding location as the most readable visualization with an average score of 5.92/7 (see Figure. 9.1a). Particularly, 6/12 participants strongly agreed, 2/12 agreed, 3/12 slightly agreed, while 1/12 disagreed. In contrast, the color texture with an integrated location with an average score of 4.67/7 and the donut chart with an overlapping location with an average of 4.50/7 were reported as less readable visualizations. In particular, looking at the colored texture scores assigned by the participants, 2/12 strongly agreed, 3/12 agreed, 2/12 slightly agreed, 1/12 was neutral, 2/12 slightly disagreed and 1/12 disagreed. Instead for the donut chart, 1/12 participants strongly agreed, 3/12 agreed, 4/12 slightly agreed, 3/12 slightly disagreed and only 1/12 participant disagreed.

Clear: In this case the scores were very close and overall all the visualizations seemed to be clear (see Figure 9.1b). The bar chart embedded around the referent received an average score of 5/7, the colored texture with an integrated location a score of 4.75/7 and the donut chart overlapping the referent a score of 4.67/7. Specifically for the bar chart, 3/12 participants strongly agreed, 2/12 agreed, 2/12 slightly agreed, 2/12 were neutral and 3/12 slightly disagreed. Looking at the colored texture scores, 2/12 participants strongly agreed, 2/12 agreed, 4/12 slightly agreed, 3/13 slightly disagreed and 1/12 disagreed.

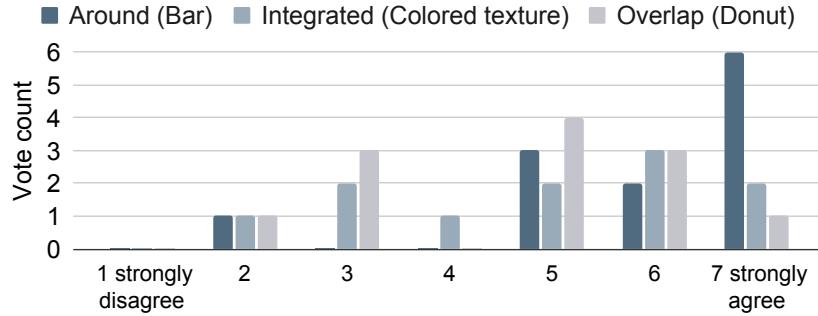
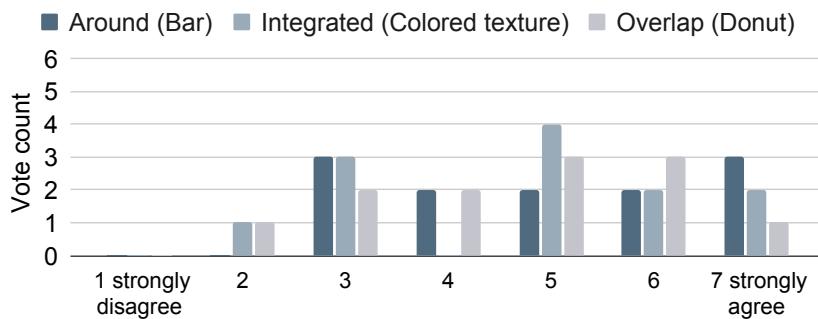
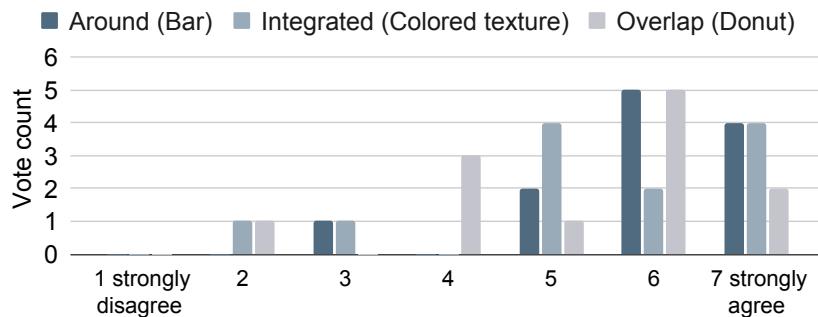
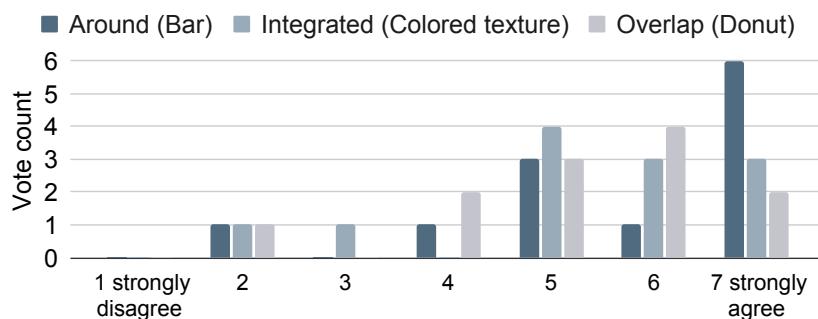
(a) Results of the *readable* likert item.(b) Results of the *clear* likert item.(c) Results of the *understandable* likert item.(d) Results of the *intuitive* likert item.

Figure 9.1: Results emerged from the readability likert-scale in the in-game questionnaire of RobotLife. The results are presented for each embedding location and for each likert item

Lastly, on the donut chart, 1/12 participant strongly agreed, 3/12 agreed, 3/12 slightly agreed, 2/12 were neutral, 2/12 slightly disagreed and 1/12 disagreed.

Understandable: The bar chart with an around embedding location resulted to be the most understandable with an average score of 5.92/7 (see Figure 9.1c). Particularly, 4/12 participants strongly agreed, 5/12 agreed, 2/12 slightly agreed and only 1/12 participant slightly disagreed. Instead, the colored texture integrated in the referent's design received an average score of 5.42/7, slightly less than the bar chart. In this regard, 4/12 participants strongly agreed, 2/12 agreed, 4/12 slightly agreed, 1/12 slightly disagreed and only 1/12 disagreed. Last, the donut chart with an overlapping embedding location received the lowest score of 5.25/7. Specifically, only 2/12 participants strongly agreed, 5/12 agreed, 1/12 slightly agreed, 3/12 participants were neutral and only 1/12 participant disagreed.

Intuitive: The bar chart with an around embedding location was voted to be the most intuitive visualization with an average score of 5.75/7 (see Figure 9.1d). 6/12 participants strongly agreed, 1/12 agreed, 3/12 slightly agreed, 1/12 was neutral and 1/12 slightly disagreed. The colored texture with an integrated embedding location turned out to be the less intuitive with an average score of 5.33/7, compared to the average score of 5.58/7 of the donut chart with an overlapping embedded location. In particular, about the colored textures, 3/12 participants strongly agreed, 3/12 agreed, 4/12 slightly agreed, 1/12 slightly disagreed and only 1/12 participant disagreed. On the donut chart, 2/12 participants strongly agreed, 4/12 agreed, 3/12 slightly agreed, 2/12 were neutral and only 1/12 disagreed.

9.2.2. Aesthetic

To evaluate the aesthetics of each visualization I used the BeauVis scale proposed by He et al (see Appendix A - Figure A.15). The 7-levels scale (1: strongly disagree, 7: strongly agree) aims at evaluating the "aesthetic pleasure" or "beauty" of a visualization. It contains 5 items including enjoyable, likable, pleasing, nice and appealing. Since BeauVis is a standard scale, I analyzed the results by following the approach proposed by the scale authors, calculating the average of each item per visualization. As shown in Figure 9.2 the colored texture with an integrated embedding location resulted to be the most beautiful visualization with an average score of 5.40/7. Specifically it resulted very enjoyable and likable. The donut chart (4.91/7) was more appreciated than the bar chart. On average the donut chart was considered more enjoyable and appealing than the bar chart that scored only 4.34/7. Although I used a quantitative measure to evaluate which visualization is more beautiful than the others, some participants expressed subjective evaluation towards

visualizations' aesthetics. The results presented here are therefore complemented by the interviews presented later in section 9.4.

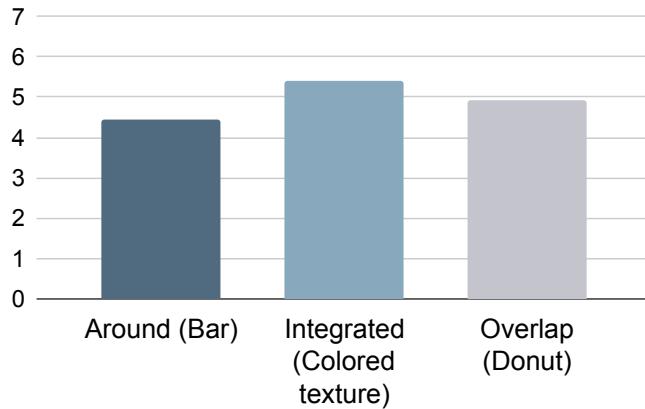


Figure 9.2: Result of the BeauVis [22] likert scale used in the in-game questionnaire to evaluate the aesthetic pleasure of the visualizations studied.

9.2.3. Summary

Overall, the bar chart with an around locations was reported as a visualization with the highest readability, while the color texture with and integrated location and the donut chart with an overlap location were less readable. The color texture's score was slightly higher than the donut one. Nevertheless, the color texture design gained the highest score in aesthetics evaluation. Instead, the bar chart was reported as the least pleasing design. Further interviews with participants were conducted for the opposite scores on readability and aesthetics, the results are presented in section 9.4.

9.3. Post-questionnaire results

After completing the gameplay of RobotLife I asked participants to complete a post questionnaire that comprised 4 ranking questions (see Appendix A - section A.9). Each question asked participants to rank the three visualizations according to a specific criteria.

Readability: This question asked how easy it was to read the visualization under motion (see Figure 9.3a). It is a confirmatory question of the in-game questionnaire. The bar chart embedded around the character was chosen as first choice by 7/12 participants, as second by 4/12 participants and as third by only 1/12 participant. The colored texture integrated in the character's design was chosen as first choice by 3/12 participants, as second by 3/12 participants and as third by 6/12 participants. The donut chart overlapping the hover bot

body was instead ranked first only by 2/12 participants, as second by 5/12 participants and as third by 5/12.

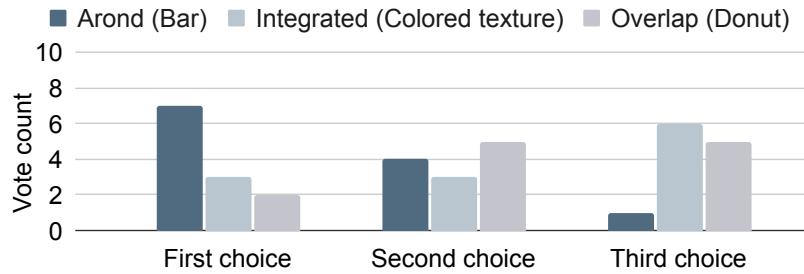
Aesthetics: This question was about how well a visualization fit in the aesthetics of the game (see Figure 9.3b). It is again a confirmatory question on the in-game questionnaire. The visualization that fitted the best was the colored texture. 9/12 participants rated the colored texture as their first choice, then 2/12 participants ranked it second and only 1/12 participant ranked it as third choice. Instead, the bar chart was preferred by only 2/12 participants as first choice, voted as second 2/12 times and 8/12 as third choice. Last, the donut chart was ranked first only 1/12 time, as second 8/12 times ad as third 3/12 times. The least preferred one was indeed the bar chart.

Supporting the game mission: This question was about how supportive were the visualizations in completing the game mission (see Figure 9.3c). The bar chart was preferred 7/12 times as first choice, voted as second by 3/12 participants and ranked third by 2/12 participants. The visualization integrated with the character's design was instead voted as first choice only 2/12 times, as second 4/12 times and as third 6/12 times. To conclude, the donut chart was ranked first by 3/12 participants, second by 5/12 participants and chosen as third 4/12 times. This results contradicted a bit the results emerging from the analysis of the game performances. In fact on average the participants won more using the donut chart rather than the bar chart or the colored texture.

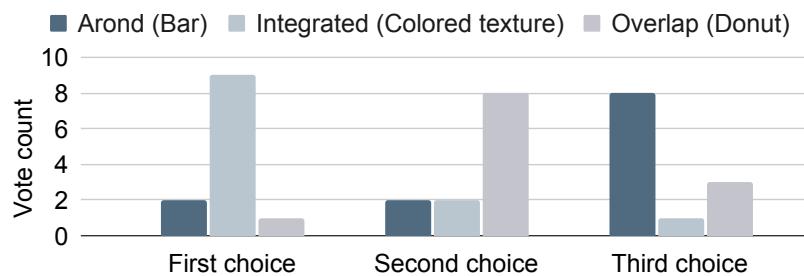
Overall experience: This question evaluated the overall preference toward a specific visualization, considering the game experience, the immersion and the usability of the visualizations (see Figure 9.3d). The bar chart was preferred as first choice by only 2/12 participants. The majority (6/12) ranked it as second choice and 4/12 participants ranked it as third choice. Overall, the colored texture was preferred by 9/12 participants. Only 2/12 participants ranked it as second choice and only 1/12 participant as third. The donut chart instead was ranked first only by 1/12 participant, as second by 4/12 and the majority ranked it third (7/12). This result suggested that there is a strong correlation between the aesthetic of the visualizations and the overall game experience and the players engagement.

9.3.1. Summary

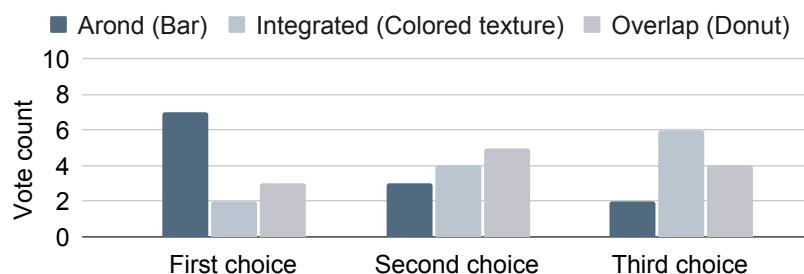
Overall, we received as similar result as before: the bar chart embedded around the character was preferred in terms of readability. The second choice was mainly the donut chart and third the colored texture. Additionally, the ranking question about the aesthetic of the visualization confirmed the results of the in-game questionnaire. The colored texture



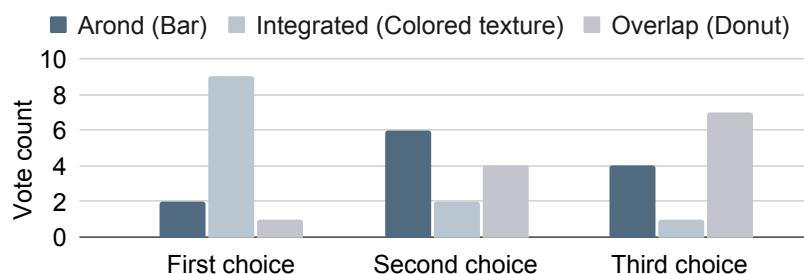
(a) Results of the question "rank the 3 visualization according to how easy they were to read while in motion".



(b) Results of the question "rank the 3 visualizations according to how well they fit into the aesthetics of the game".



(c) Results of the question "rank the 3 visualizations according to how well they supported you completing your mission".



(d) Results of the question "rank the 3 visualizations according to how supportive you found them for a positive overall game experience."

Figure 9.3: Results emerged from the post-questionnaire completed by the pilot participants after playing RobotLife.

was perceived as the visualization that better fitted the aesthetics of the game, followed by the donut chart and last by the bar chart. In terms of how supporting each visualization was to complete the game mission, the bar chart was ranked as the first, followed by the donut chart and the colored texture. Within the scope of the overall experience, participants reported that they preferred the colored texture integrated in the character's design rather than the bar chart and the donut chart.

9.4. Interview results

In order to better understand how and why different embedding locations influenced the visualizations readability as well as the impact they had on the game character aesthetic, we interviewed participants at the end of the experiment. Particularly, we asked questions including:

1. How did the movement of the hover bots influence how well you could read the visualizations?
2. How did the movement of your character influence how well you could read the visualizations?
3. What strategies did you create to counteract any influence of motion?
4. Do you think that the fact that the three visualization were embedded in the robots in three different location, influenced how well you could read those visualization?
5. How difficult did you find the game overall?

We also asked participant to explain their rankings concerning the post-questionnaire to gain more visualization specific insights. During the interviews, two interviewers took notes about the participants answers. Subsequently, those notes were coded into 6 categories.

9.4.1. Readability

This category explains why participants felt a specific visualization was easy or difficult to read. The bar chart embedded around the character was considered easy to read by 4/12 participants because of its representation (an horizontal line). Also, 2/12 participants declared that it was easy to read because they used a strategy as they divided the bar chart into 3 slices and 66% was easy to identify. Furthermore, 4/12 participants declared to be more familiar with the bar because it is the mainstream way to represent health in video games. Moreover, 2/12 participants agreed that the bar chart was more readable because of its embedding location (around the character). Particularly, the over head

position made the visualization more visible and the easier to read. On the contrary, 3/12 participants found difficult to spot the 66% with this visualization.

Regarding to the colored texture integrated in the character's design, the main difficulty reported by the participants was the hover bot animation. While in motion, the hover bots inclined their top part by an angle of 45°, covering partially the eye-ball back side. This feature made difficult to see the eye-ball from the back of the robots so participants reported that it was difficult for them to see some proportions and distinguish good from evil robots. Moreover, 2/12 participants stated that their unfamiliarity with the colored texture made negative influence on visualization's readability. They stated that they felt the color brightness had impact on readability as well. On the contrary, 1/12 participant reported that it was easy to read the colored texture by a glance, while another 1/12 participant said it was easier to read from the distance.

The donut chart overlapping the body of the character could be quickly estimated because they used a strategy to divide the donut per 25%, and 66% is in middle of 50% and 75%. 1/12 participant declared that the representation (a circularity) of the donut chart made the visualization the easiest to read but on the contrary, 4/12 participants declared that the shape made it harder to read. Also, this visualization was reported as the most difficult one. According to 3/12 participants who reported this, it was because the donut chart required more mental effort to calculate proportions, especially for the percentages closed to 66%.

9.4.2. Aesthetic

Participants expressed their subjective opinion about why they thought a visualization was beautiful. The majority of our participants (5/12) reported that the bar chart was not a pleasing design. Among them, 2/12 participants criticized that the bar chart visualization was too mainstream, while 3/12 participants criticized that the embedding location reduced the beauty of the visualization. The over head position was not directly integrated in the character and therefore it was perceived as an external element that did not match the game aesthetic. On the contrary, only 2/12 participants declared they preferred the bar chart because of their familiarity.

The colored texture was perceived as the most beautiful because it was directly integrated in the character's design. The majority of the participants (8/12) declared that the embedded design made the visualization more immersive and added some challenge to the game experience. Nevertheless, 1/12 participant considered it less aesthetic because of their unfamiliarity, while 1/12 participant said they did not like the embedded design

because in some dark areas the color itself cannot stand out because of the lightness rendering.

The donut chart instead was considered beautiful by 4/12 participants because of its embedded location – directly surrounded the robot’s eyes and matched to the robot design. In this case, the preferences for this visualization referred specifically to the character design and may vary with other robot design. On the contrary, 1/12 participant said that the donut chart was not beautiful because hard to read, undermining the beauty of the character.

9.4.3. Embedding Locations

Since the objective of the study was to evaluate the influence of different embedding locations, we particularly focused on asking participants how this contextual factor influenced the visualization designs. According to 4/12 participants, the around location, specifically over the head of the robots, had positive impact on readability. It influenced positively the visualization design because it made the visualization more visible. Furthermore, this embedding location was considered by the 4/12 participants to be the most familiar to them. Instead, 2/12 participants had negative comments on this embedding location because it required extra space that could lead to confusion when robots overlap with each other.

3/12 participants reported that they preferred integrating the visualization in the character because it could fit to any character’s design. Also, 2/12 participants appreciated this embedding location because it saved space, without adding any extra element to the game environment. On the contrary, 1/12 participant stated that the integrated position could confuse the character design such as partially occluding the robot’s eyes. Also, 1/12 participant said this locations can easily be overlapped by other objects and props in the game context, which physically reduced the visualization’s readability.

Last, overlapping a visualization on the robot body was appreciated by 4/12 participants because the particular shape of the donut chart matched the robot design (particularly the eye-ball shape). 2/12 participants also liked it because it saved space and directly indicated the robot. According to 1/12 participant, this embedding location made the visualization more readable while another participant stated that it was easily covered by other objects in the scenario and therefore this embedding locations was worse than other representations like bar chart.

9.4.4. Hover bot motion

Focusing the attention on motion factors occurring while playing RobotLife, we asked participants how the movement of the hover bot influenced their ability to read the visualizations. 4/12 participants declared that motion had generally a negative impact on their readability. Moreover, 5/12 said that robots were easily occluded by other objects in the map because, by moving around freely, the robots tend to go behind objects or inside different rooms. Also, speed was perceived as an influencing factor by 1/12 participant. Instead, 1/12 participant did not experience any influence of motion, saying that this factor had no impact on their estimation. Only one specific comment was given regarding to the bar chart visualization. Our participant declared that the over head position helped tracking the robot trajectory, making reading the visualization easier under motion.

9.4.5. Self-motion

Self motion is the ability of the player to control the protagonist of the game, move around the scenario or rotate the camera and its field of view. This category then represents the predictable and controlled motion factors. Generally, the impact of this factor depends on how experienced our participants were - experienced game players said they did not feel self-motion had negative impact while novice reported they cannot do the task while moving. 8/12 participants had to stop to read the visualizations, trying to remove this effect. On the other hand, 3/12 participants exploited it to get closer to the robots and see better their health visualizations. So controlled movement in this case improved the visibility of the visualizations. 1/12 participant instead experience no influence.

9.4.6. Strategies

While playing RobotLife, some participants developed interesting strategies to counterbalance the impact of motion factors and better estimate the visualizations. The possibility to zoom-in by aiming with the game weapon was used by 3/12 participants to have a closer look at the health visualizations. 4/12 participants always waited for the robots to stop before shooting at them, to be sure about their health level and read while the robots were still. Instead, going closer was the strategy adapted by 3/12 participants to read better and reduce the impact of autonomous motion factors. Only 1/12 participant declared the use of eye tracking to predict the robot movement while 2/12 participants had the need to navigate around the map and compare health levels of different robots before shooting at them.

9.5. Discussion

Results showed that the bar chart embedded around the character was the most readable visualization under motion in the proposed video game, with the reason of its shape and participants' familiarity. The embedding location around the character allowed to avoid occlusion, which improves the visualizations' readability from a distance. The design of the bar chart with around location was not particularly affected by motion factors. On the one hand, the bar chart supported players in tracking the movement of the robots. On the other hand, the around location indicated the position of the game characters. Nevertheless, this design was not appreciated in terms of aesthetics. The bar chart was considered too mainstream. The around location was criticized because it occupied extra space other than the character itself and was not directly integrated with it.

Instead, the colored texture integrated in the character design resulted in the most beautiful visualization. The integrated design was particularly appreciated because it enhanced the game immersion and was considered more challenging and, therefore, more fun. The integrated design was also criticized because the rendering had a slightly impact on its effect of the red color: the color contrast changes with context lightness. Therefore, the visualization resulted less visible and difficult to read.

The donut chart overlapping the character's body seemed to be a good trade-off between readability and aesthetics. Dividing a donut chart into quarters helped participants to read the visualization, especially under motion. Furthermore, the similarity between the donut shape and the hover bot eye-ball resulted in a aesthetically pleasing design.

Regarding the impact of motion factors, they generally had a negative impact on the visualizations' readability. The autonomous movement of the robots particularly impacted the visualizations integrated and overlapping the game characters due to occlusions caused by the robot moving behind other game objects. Some of the participants felt that they needed to wait for the robots to stop before estimating their health level. It seems participants needed to develop a counterbalancing strategy eliminating the impact of motion factors. However, this strategy may be changed if our participants were all experienced game players. The controlled movement derived from the player commands also influenced the visualization readability. In fact, the majority of the players decided to stop before estimating each visualization. This result was also due to the level of game playing expertise of our participants. Finally, all the participants declared that the game was fun and easy. Also, the less experienced players learned the game commands and mechanism very quickly, proving that the game design was consistent and adaptable to different levels of expertise.

10 | Considerations and Limitations

After giving the experiment's results above, here I first present my design considerations for visualizations in motion in the context of video games. Although we conducted a pilot study rather than a formal experiment, we still discuss the limitations that can be improved in future work.

10.1. Design Considerations

From our post-interview, we noticed that participants had a willing tendency to do a trade-off between representations and readabilities. Here, we presents our design considerations according to our post-interview results.

- **Use familiarity reasonably**

Data in video games must be read at a glance. Participants preferred to get needed information from moving visualization in a short time. A widely used design is more familiar to them and reported as less reading time consuming. Moreover, using a mainstream visual representation can help players perceive faster. Instead, an unfamiliar visualization will require more explanation and reaction time. When designing a visualization for specific data, video game visualization practitioners should consider if this data needs to be immediately consumed.

- **Enhance or reduce immersion**

The general tendencies of video games nowadays is pushing towards diegetic interfaces and data representation. Diegetic elements enhances the immersion of the game and therefore result in more engaging and fun game for the players. At the same time, interpreting visualizations integrated in the game element's design usually requires more effort and attention. Those visualization are subject to the specific element's design characteristics, are directly altered by the lighting and rendering conditions and highly depends on the game element's position and orientation. Participants

said that an immersive visualization design could improve their pleasure. They also reported that an immersive element could be easily mixed with an external entity. Thus, a trade-off should be made between immersion and remarkability. Especially when an elemental such as a warning should be seen immediately and stand out.

- **Create tailored designs**

Not every visual representation, color, embedding location and more, can fit perfectly a game element design. Thus, for a specific game element, matching the visualization with a good embedding location is the key to balance its readability and aesthetics. If the game element is particularly difficult to pair with a visualization, or the data that need to be embedded can only be represented with a specific visual representation, it is always better to embed the visualization *around the data referent* with respect to other embedding locations. Participants commented that the donut chart was a good and immersive design when being overlapped around the robot's eyes. However, they could imagine that the donut chart would become an unsuitable design if it was put over the head of the robot.

- **Make a good color selection**

The color palette of each video game is unique. Using colors that don't match the video game aesthetics can reduce the video game quality and, subsequently, the players' engagement. A visible visualization requires high color contrast with its background. Aside, colors are also one of the best ways to encode meanings in video games. For example, it is standard to use green to represent good things and red to show enemies or danger. A correct color helps players to know faster what a visualization is about. Therefore, it is significant to make a good color selection.

- **Consider the visibility conditions**

The camera orientation is one of the first decisions a game designer makes during the prototyping phase. Due to the chosen camera perspective, a first person, third person, top-down, or a 2D game has a perceptual difference. Different camera orientations would lead to overlapping between game entities. It is necessary to consider whether a visualization should always be visible in the foreground or can be occluded or hidden by other elements. This design consideration depends on the temporal continuity of the data. That means if the data must be displayed all the time, its visual representation should not be impeded. Some specific transitions should be made, such as changing embedding locations, switching colors, or zooming in and out.

- **Take motion factors into account**

Designing visualization in motion is completely different from creating stationary visualization. The impact of relative movement between visualizations and viewers cannot be neglected. A reasonable way to make an effective design is to smooth the motion by reducing the speed or simplifying the movement trajectory. The use of hieroglyphics such as text and numbers is less recommended, while the use of totems such as shapes and symbols is encouraged.

10.2. Limitations

A main limitation of this research is that, as a pilot study, I do not have a large number of participants. Although my result may vary from a future formal experiment it is still sufficient and able to give a prediction of the real experiment. Another limitation is, that due to the limited time, I did not have a chance to recruit experienced game players. Only a few of my participants were game playing experts. The lack of experience in video games may have influenced the gaming performance as well as the overall gaming experience. It may also cause biases in part of the answers of the participants. Also, more experienced gamers could have provided more valuable insights since they are aware of the current practises in the analyzed scenario and they are used to certain types of visualizations (i.e. bar chart for health visualization).

Another limitation of the study is that we have to bind visual representations with embedding locations. While the main goal of the study was studying the impact of embedding locations, in the context of video game it is unrealistic to take into account only the embedding position without considering the aesthetics and rationality of the visual representation. Many participants appreciated one visualization more than the others not only because of the embedding location but because a good visual representation was displayed at a suitable position. Therefore, embedding locations and visual representations should not be discussed separately but combined to provide more practical results.

To tackle the limitations above, in future formal experiments, enough experienced game players should be recruited. To conclude, the contextual factors influencing the three visualizations in the proposed pilot study are multiple and difficult to control at the same time. To produce a more accurate and targeted experiment to evaluate one factor at the time, the video game experience should be altered, removing fundamental features of the gameplay. I decided not to do in order to provide valuable results applicable in a real video-game scenario.

11 | Conclusions

In the context of video games various contextual factors, such as embedded locations, have to be considered while analyzing situated visualization in motion. By conducting a systematic review of the current design practices in 50 video games, I acquired extensive knowledge on how data was embedded with different game elements and which are the main contextual factor to consider to describe a situated visualization. Moreover, I learned how different types of data are represented in games. Specifically I explored how character's health values (quantitative data) are visualized as well as the game element types (categorical data). Results showed that the type of representations used depended on the game genre, and some genres were more visualizations-prone than others. RPGs, strategy, and war games are the most visualization-prone genres. The most used visual representations were signs, bar charts, and labels with both numbers and text, and they were mainly embedded around game characters. To help players read the visualizations at a glance, a single piece of information is visualized. Colors became a powerful method to convey information at a glance. Finally, results showed that by interacting with the game context, either moving around or just rotating the game camera, players are the leading cause of visualizations' motion. To conclude, I developed RobotLife and run a pilot study to evaluate the impact of embedding locations on visualizations' readability and game character's aesthetics. The designed health visualization embedded in RobotLife were meant to highlight three different embedding locations with their own features and limitations. Results showed that:

- Embedding visualizations around characters supported the visualization readability.
- Integrating visualizations in the character's design increased the visualization aesthetics.
- Embedding visualization overlapping the character is a good trade-off between readability and aesthetics but needs a the visualization design matching the game character design.
- Motion factors had a negative impact on the visualizations' readability.

11.1. Future Work

Since the study presented in this thesis was conducted in the form of an extensive pilot, a formal experiment is required to gather more informative data. Useful suggestions can be taken into considerations from the presented pilot study, in order to improve the final study design, addressing the limitations highlighted in section 10.2. Moreover, the topic of visualizations in motion in video games can be furtherly explored by taking into account other design factors, such as different visual representations for the same type of data, or the use of specific colors to draw the attention of the players. Since the implementation of a second game level of RobotLife was already in place, a new study can be designed exploiting the level mechanics implemented. Particularly, the second level described in section 7.5 allows to evaluate the impact of different visual representation on situated visualization's readability, exploiting three designs of visualizations of game element types. Thus, an other empirical study will come soon.

Bibliography

- [1] J. R. Babu. Video game huds: Information presentation and spatial immersion. Master's thesis, Rochester Institute of Technology, 2012.
- [2] A. Bezerianos and P. Isenberg. Perception of visual variables on tiled wall-sized displays for information visualization applications. *IEEE Transactions on Visualization and Computer Graphics*, 2012.
- [3] B. Bowman, N. Elmquist, and T. Jankun-Kelly. Toward visualization for games: Theory, design space, and patterns. *IEEE Transactions on Visualization and Computer Graphics*, 2012.
- [4] A. Brooksby. Exploring the representation of health in videogames: A content analysis. *Cyberpsychology and behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, 2008.
- [5] F. Buccieri, L. Yao, and P. Isenberg. Visualization in Motion in Video Games for Different Types of Data. In *Journée Visu*, 2022.
- [6] F. Buccieri, L. Yao, and P. Isenberg. Situated Visualization in Motion for Video Games. In *In Proceeding of the Eurographic Conference on Visualization*, 2022.
- [7] C. Chen. Information visualization: Beyond the horizon. *Information Visualization*, 2002.
- [8] S. Claes and A. Vande Moere. Street infographics: Raising awareness of local issues through a situated urban visualization. In *2nd ACM International Symposium on Pervasive Displays*, 2013.
- [9] D. Clarke and P. R. Duimering. How computer gamers experience the game situation: A behavioral study. *Computers in Entertainment*, 2006.
- [10] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 1984.

- [11] D. Conroy, P. Wyeth, and D. Johnson. Understanding player threat responses in fps games. In *Proceedings of The 9th Australasian Conference on Interactive Entertainment: Matters of Life and Death*. Association for Computing Machinery, 2013.
- [12] J. Deleuze, M. Christiaens, F. Nuyens, and J. Billieux. Shoot at first sight! first person shooter players display reduced reaction time and compromised inhibitory control in comparison to other video game players. *Computers in Human Behavior*, 2017.
- [13] E. Dickmark. The use of colour in the game journey : Case study, 2015.
- [14] M. S. El-Nasr and S. Yan. Visual attention in 3d video games. In *Proceedings of the ACM International Conference on Advances in Computer Entertainment Technology*. Association for Computing Machinery, 2006. ISBN 1595933808.
- [15] H. Elias. *First Person Shooter: The Subjective Cyberspace*. PhD thesis, University of Beira Interior, 2009.
- [16] N. A. M. ElSayed, B. H. Thomas, R. T. Smith, K. Marriott, and J. Piantadosi. Using augmented reality to support situated analytics. In *IEEE Virtual Reality*, 2015.
- [17] C. Fabricatore, M. Nussbaum, and R. Rosas. Playability in action videogames: A qualitative design model. *Human-computer Interaction*, 2002.
- [18] E. Fagerholt and M. Lorentzon. Beyond the hud - user interfaces for increased player immersion in fps games. Master's thesis, Chalmers University of Technology, 2009.
- [19] Footovision. Reveal the true power of sports analytics, 2021. <https://www.footovision.com>.
- [20] C. Gittens and P. Gloumeau. Does a segmented health bar affect a player's preference for a game? a pilot study. In *IEEE Games Entertainment Media Conference (GEM)*, 2015.
- [21] L. Harrison, F. Yang, S. Franconeri, and R. Chang. Ranking visualizations of correlation using weber's law. *IEEE Transactions on Visualization and Computer Graphics*, 2014.
- [22] T. He, P. Isenberg, R. Dachselt, and T. Isenberg. Beauvis: A validated scale for measuring the aesthetic pleasure of visual representations. In *IEEE Visualization Conference*, 2022.

- [23] N. Hoobler, G. Humphreys, and M. Agrawala. Visualizing competitive behaviors in multi-user virtual environments. In *IEEE Visualization*, 2004.
- [24] T. Horbiński and K. Zagata. Map symbols in video games: the example of “valheim”. *KN - Journal of Cartography and Geographic Information*, 2021.
- [25] T. Hynninen. First-person shooter controls on touchscreen devices: a heuristic evaluation of three games on the ipod touch. Master’s thesis, University of Tampere, 2012.
- [26] A. Islam, A. Bezerianos, B. Lee, T. Blascheck, and P. Isenberg. Visualizing Information on Watch Faces: A Survey with Smartwatch Users. In *IEEE Visualization Conference Short Papers*, 2020.
- [27] P. Isokoski and B. Martin. Performance of input devices in fps target acquisition. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology*, 2007.
- [28] E. Joosten, G. van Lankveld, and P. Spronck. Colors and emotions in video games. *11th International Conference on Intelligent Games and Simulation*, 2010.
- [29] M. Karlsson. Information visualisation in games: How do you know what you know? Master’s thesis, Umeå University, Faculty of Social Sciences, Department of Informatics., 2016.
- [30] A. Kirk. *Data Visualisation: A Handbook for Data Driven Design*. SAGE Publications Ltd, 2016.
- [31] N. Kong, J. Heer, and M. Agrawala. Perceptual guidelines for creating rectangular treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 2010.
- [32] Konnikova, Maria. Why gamers can’t stop playing first-person shooters. <https://www.newyorker.com/tech/annals-of-technology/why-gamers-cant-stop-playing-first-person-shooters>, 2013.
- [33] J. Looser, A. Cockburn, and J. Savage. On the validity of using first-person shooters for fitts’ law studies. *People and Computers XIX*, 2005.
- [34] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 2021.
- [35] Q. Marre, L. Caroux, and J.-C. Sakdavong. Video game interfaces and diegesis: The impact on experts and novices’ performance and experience in virtual reality. *International Journal of Human-Computer Interaction*, 2021.

- [36] J. Martinez, E. Adi, and A. Prima. A study of colour as an attribute that intensifies user's engagement in game plays. *International Journal of Multimedia & Its Applications*, 2012.
- [37] B. Medler. Generations of game analytics, achievements and high scores. *Eludamos: Journal for Computer Game Culture*, 2009. URL <https://www.eludamos.org/index.php/eludamos/article/view/vol3no2-4>.
- [38] Metacritic. Movie Reviews, TV reviews, game reviews, and music reviews, 2022. <https://www.metacritic.com/>.
- [39] Metacritic. Call of duty for pc reviews - metacritic, 2022. <https://www.metacritic.com/game/pc/call-of-duty>.
- [40] Metacritic. Cyberpunk 2077 for pc reviews - metacritic, 2022. <https://www.metacritic.com/game/pc/cyberpunk-2077>.
- [41] A. S. Originals. Snaps prototype | sci-fi / industrial, 2022. <https://assetstore.unity.com/packages/3d/environments/sci-fi/snaps-prototype-sci-fi-industrial-136759>.
- [42] M. Peacocke, R. J. Teather, J. Carette, I. S. MacKenzie, and V. McArthur. An empirical comparison of first-person shooter information displays: Huds, diegetic displays, and spatial representations. *Entertainment Computing*, 2018.
- [43] Z. Pousman and J. Stasko. A taxonomy of ambient information systems: Four patterns of design. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. Association for Computing Machinery, 2006.
- [44] L. Quevedo, J. Aznar-Casanova, D. Merindano, and J. Solé. A task to assess dynamic visual acuity and a valuation of the stability of its measurements. *Psicologica*, 2010.
- [45] L. Quevedo, J. Aznar-Casanova, and J. A. Da Silva. Dynamic visual acuity. *Trends in Psychology*, 2018.
- [46] R. Roth, K. Ross, and A. MacEachren. User-centered design for interactive maps: A case study in crime analysis. *International Journal of Geo-Information*, 2015.
- [47] SportsDynamics. Reveal the invisible dynamics behind the game, 2022. <https://sportsdynamics.eu>.
- [48] U. Technologies. Fps microgame. <https://learn.unity.com/project/fps-template>, 2022.

- [49] U. Technologies. Unity - manual: Navigation system in unity, 2022. <https://docs.unity3d.com/Manual/nav-NavigationSystem.html>.
- [50] U. Technologies. Real-time development platform | 3d, 2d, vr & ar engine, 2022. <https://unity.com>.
- [51] U. Technologies. Unity - manul: Creating and using scripts, 2022. <https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html>.
- [52] U. Technologies. Probuilder, 2022. <https://unity.com/features/probuilder>.
- [53] B. Thomas, G. Welch, P. Dragicevic, N. Elmquist, P. Irani, Y. Jansen, D. Schmalstieg, A. Tabard, N. Elsayed, R. Smith, and W. Willett. Situated analytics. *Immersive Analytics*, 2018.
- [54] A. Vande Moere and D. Hill. Designing for the situated and public visualization of urban data. *Journal of Urban Technology*, 2012.
- [55] R. Vicencio-Moreira, R. L. Mandryk, C. Gutwin, and S. Bateman. The effectiveness (or lack thereof) of aim-assist techniques in first-person shooter games. In *Proceedings of the Conference on Human Factors in Computing Systems*. Association for Computing Machinery, 2014. ISBN 9781450324731.
- [56] C. Ware. Visual thinking for information design. In *Visual Thinking for Information Design*. Morgan Kaufmann, 2021.
- [57] D. Weiskopf. On the role of color in the perception of motion in animated visualizations. In *IEEE Visualization Conference*, 2004.
- [58] S. White. Interaction and presentation techniques for situated visualization. *Columbia University*, 2009. https://ui.st.acm.org/archive/adjunct/2008/pdf/doctoral_symposium/paper151.pdf.
- [59] S. White and S. K. Feiner. Sitelens: situated visualization techniques for urban site visits. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009.
- [60] Wikipedia contributors. Bicycle counter — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Bicycle_counter&oldid=1073530998, 2022.
- [61] Wikipedia contributors. National debt clock — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=National_Debt_Clock&oldid=1092638477, 2022.

- [62] Wikipedia contributors. Health (game terminology) — Wikipedia, the free encyclopedia. [https://en.wikipedia.org/w/index.php?title=Health_\(game_terminology\)&oldid=1073646330](https://en.wikipedia.org/w/index.php?title=Health_(game_terminology)&oldid=1073646330), 2022.
- [63] W. Willett, Y. Jansen, and P. Dragicevic. Embedded data representations. *IEEE Transactions on Visualization and Computer Graphics*, 2016.
- [64] L. Yao, A. Bezerianos, and P. Isenberg. Situated visualization in motion. In *Posters of the IEEE Visualization Conference*, 2020.
- [65] L. Yao, A. Bezerianos, R. Vuillemot, and P. Isenberg. Situated Visualization in Motion for Swimming. In *Journée Visu*, 2022.
- [66] L. Yao, A. Bezerianos, R. Vuillemot, and P. Isenberg. Visualization in motion: A research agenda and two evaluations. In *IEEE Transactions on Visualization and Computer Graphics*, 2022.
- [67] V. Zammitto. Visualization techniques in video games. *Electronic Visualisation and the Arts Conference*, 2008.

A | Appendix A

A.1. Systematic review: Selected Video Games

	Genre	Video game Title	Platform	Year
1	Action	Gear of war	Xbox One	2016
2	Action	Call of Duty: Black Ops 4	PlayStation 4	2018
3	Action	Battlefield V	PlayStation 4	2018
4	Adventure	Detroit: Become Human	PlayStation 4	2018
5	Adventure	Subnautica: Below Zero	Xbox Series S	2021
6	Adventure	Sam and Max Save the World - Remastered	PC	2020
7	Fighting	Super Smash Bros	Wii U	2014
8	Fighting	Deadpool	PC	2013
9	Fighting	PlayStation All-Stars Battle Royale	PlayStation 3	2012
10	FPS	RAGE 2	PlayStation 4	2019
11	FPS	Borderlands 3	PlayStation 4	2019
12	FPS	Halo Infinite	Xbox Series X	2021
13	Flight/Flying	Microsoft Flight Simulator	Xbox Series S	2021
14	Flight/Flying	The Falconeer	PC	2020
15	Flight/Flying	Skies of Fury DX	Switch	2018
16	Party	Mario Party 9	Wii	2012
17	Party	Nintendo Land	Wii U	2012
18	Party	Go Vacation	Wii	2011
19	Platformer	Donkey Kong Country: Tropical Freeze	Wii U	2014
20	Platformer	Super Mario 3D World	Wii U	2013
21	Platformer	Terraria	PC	2011
22	Racing	Mario Kart Tour	iOS	2019
23	Racing	Gran Turismo 7	PlayStation 5	2022
24	Racing	Forza Horizon 5	Xbox Series S	2021
25	RTS	Orcs Must Die! 3	PC	2021
26	RTS	Pikmin 3 Deluxe	Switch	2020
27	RTS	Crusader Kings III	PC	2020
28	RPG	Cyberpunk 2077	PC	2020
29	RPG	Wolcen: Lords of Mayhem	PC	2020
30	RPG	Biomutant	PlayStation 4	2021
31	Simulation	MechWarrior 5: Mercenaries	PC	2019
32	Simulation	Dirt 5	PlayStation 5	2020
33	Simulation	F1 2020	PlayStation 4	2020
34	Sport	NBA 2K21	PlayStation 5	2020
35	Sport	Madden NFL 21	PlayStation 4	2020
36	Sport	MLB The Show 20	PlayStation 4	2020
37	Strategy	Phoenix Point	PC	2019
38	Strategy	Gears Tactics	PC	2020
39	Strategy	A Total War Saga: TROY	PC	2020
40	TPS	Aliens: Fireteam Elite	PC	2021
41	TPS	Returnal	PlayStation 5	2021
42	TPS	Outriders	PC	2021
43	Turn-Based Strategy	King's Bounty II	PlayStation 5	2021
44	Turn-Based Strategy	Sakura Wars	PlayStation 4	2020
45	Turn-Based Strategy	XCOM: Chimera Squad	PC	2020
46	Wargames	Worms Battlegrounds	PlayStation 4	2014
47	Wargames	Toy Soldiers	PC	2012
48	Wargames	Sengoku	PC	2011
49	Wrestling	WWE 2K14	PlayStation 3	2014
50	Wrestling	Fire Pro Wrestling	Xbox 360	2012

Table A.1: List of selected games for the systematic review conducted.

A.2. Systematic review: Data collected

Index	Game's name	Genre	Link	Type of data dimensions	Frequency of appearance	On Focus	Representative used	Color appearance	Foreground	Background	Data referent	Embedding locations	Movement autonomy	
1	Gears of War 4	Action	https://gearsofwar.com	Spatial	1	Frequent	No	Sign	Thematic	No	Varying with vis position	Character	Around	controlled
2	Gears of War 4	Action	https://gearsofwar.com	Nominal; Quantitative	1	Frequent	Yes	Silhouettes	Hidden	Yes	Varying with vis position	Character	Around	controlled
3	Call of Duty: Black Ops 4	Action	https://www.callofduty.com/en/blackops4	Nominal; Quantitative	2	Frequent	No	Areas. Color	Thematic	No	Fixed	Location	Around	controlled
4	Call of Duty: Black Ops 4	Action	https://www.callofduty.com/en/blackops4	Nominal; Quantitative	1	Moderate	Yes	Overlapped	Salient	Yes	Varying with vis position	Character	Over	both
5	Call of Duty: Black Ops 4	Action	https://www.callofduty.com/en/blackops4	Nominal; Quantitative	2	Always Visible	No	Sign	Thematic	Yes	Varying with vis position	Location	Around	controlled
6	Call of Duty: Black Ops 4	Action	https://www.callofduty.com/en/blackops4	Nominal; Quantitative	1	Moderate	No	Circular bar	Thematic	Yes	Varying with vis position	Location	Around	controlled
7	Call of Duty: Black Ops 4	Action	https://www.callofduty.com/en/blackops4	Nominal; Quantitative	2	Frequent	No	Bar	Salient	Yes	Varying with vis position	Character	Around	both
8	Call of Duty: Black Ops 4	Action	https://www.callofduty.com/en/blackops4	Nominal	1	Frequent	No	Label with Text	Salient	Yes	Varying with vis position	Character	Around	both
9	Battlefield V	Action	https://www.ea.com/games/battlefield/v	Spatial	1	Frequent	No	Sign	Hidden	No	Varying with vis position	Character	Around	controlled
10	Battlefield V	Action	https://www.ea.com/games/battlefield/v	Spatial; Nominal	2	Frequent	No	Sign	Thematic	Yes	Varying with vis position	Character	Around	both
11	Battlefield V	Action	https://www.ea.com/games/battlefield/v	Nominal	1	Frequent	No	Label with Text	Thematic	Yes	Varying with vis position	Character	Around	both
12	Battlefield V	Action	https://www.ea.com/games/battlefield/v	Quantitative	1	Rare	No	Bar	Thematic	Yes	Varying with vis position	Character	Around	both
13	Battlefield V	Action	https://www.ea.com/games/battlefield/v	Nominal; Spatial	2	Always Visible	No	Sign	Thematic	Yes	Varying with vis position	Location	Around	controlled
14	Detroit: Become Human	Adventure	https://www.smashbox.com/en_gb/index.html?categoryid=become-human	Nominal	2	Frequent	No	Sign	Thematic	Yes	Varying with vis position	Object	Over	controlled
15	Super Smash Bros	Fighting	https://en.wikipedia.org/wikibooks/Games/GB/index.html	Quantitative	1	Always Visible	No	Label with Text	Thematic	Yes	Varying with vis position	Character	Around	autonomous
16	Deadpool	Fighting	https://borderlands.com/en/FR/video-game/	Nominal	1	Frequent	Yes	Bar	Thematic	No	Varying with vis position	Character	Around	both
17	Deadpool	Fighting	https://borderlands.com/en/FR/video-game/	Spatial	1	Frequent	Yes	Sign	Thematic	No	Varying with vis position	Object	Over	controlled
18	Borderlands 3	FPS	https://borderlands.com/en/FR/	Nominal	1	Frequent	No	Label with Text	Salient	No	Varying with vis position	Character	Around	both
19	Borderlands 3	FPS	https://borderlands.com/en/FR/	Spatial	1	Frequent	No	Sign	Thematic	Yes	Varying with vis position	Object	Around	both
20	Microsoft Flight Simulator	Flight/Flying	https://flightsimulatorto.com/	Spatial	1	Frequent	No	Sign	Thematic	Yes	Varying with vis position	Object	Around	controlled
21	The Falconeer	Flight/Flying	https://www.thefalconeer.com/	Spatial	1	Frequent	Yes	Circle	Salient	Yes	Varying with vis position	Character	Over	both
22	The Falconeer	Flight/Flying	https://www.nintendogs.comstore/product/skates-glunz-dx-switch/	Spatial; Nominal; Quantitative	1	Frequent	Yes	Circular bar	Hidden	No	Varying with vis position	Character	Over	both
23	Skies of Fury DX	Flight/Flying	https://www.nintendogs.comstore/product/skates-glunz-dx-switch/	Spatial; Nominal; Quantitative	3	Always Visible	No	Sign	Salient	Yes	Varying with vis position	Character	Around	both

Figure A.1: Systematic Review Data - Page 1.

A | Appendix A

24	Mario Party 9	Party	https://www.nintendo.com/games/wii/mario-party-9.html	Nominal	1	Frequent	No	Color Overlapped	Thematic	Yes	Varying with vis position	Character	Over	controlled
25	Nintendo Land	Party	https://www.nintendo.com/games/wiiu/nintendo-land-2.html	Quantitative	1	Rare	Yes	Pictograph	Thematic	Yes	Fixed	Location	Integrated	controlled
26	Nintendo Land	Party	https://www.nintendo.com/games/wiiu/nintendo-land-2.html	Quantitative	1	Frequent	Yes	Bar	Thematic	Yes	Varying with vis position	Character	Around	both
27	Nintendo Land	Party	https://www.nintendo.com/games/wiiu/nintendo-land-2.html	Quantitative	1	Frequent	No	Circular bar	Thematic	Yes	Fixed	Character	Integrated	controlled
28	Go Vacation	Party	https://www.nintendo.com/games/wiiu/nintendoswitch-go-vacation.html	Spatial	2	Frequent	No	Sign	Salient	Yes	Varying with vis position	Location	Around	controlled
29	Donkey Kong Country: Tropical Freeze	Platformer	https://www.nintendo.com/games/wiiu/donkey-kong-country-tropical-freeze.html	Nominal	1	Frequent	No	Color Overlapped	Thematic	No	Varying with vis position	Character	Over	controlled
30	Super Mario 3D World	Platformer	https://www.nintendo.com/games/wiiu/super-mario-3d-world.html	Nominal	1	Frequent	No	Color Overlapped	Thematic	No	Varying with vis position	Character	Over	controlled
31	Super Mario 3D World	Platformer	https://www.nintendo.com/games/wiiu/super-mario-3d-world.html	Quantitative	1	Frequent	No	Label with Numbers	Thematic	Yes	Varying with vis position	Character	Around	autonomous
32	Mario Kart Tour	Racing	https://www.nintendo.com/games/mariokarttour.html	Spatial; Nominal	2	Frequent	No	Sign	Salient	Yes	Varying with vis position	Character	Around	both
33	Mario Kart Tour	Racing	https://www.nintendo.com/games/mariokarttour.html	Spatial; Nominal	2	Frequent	No	Label with Text	Thematic	Yes	Varying with vis position	Character	Around	both
34	Gran Turismo 7	Racing	https://www.granturismo.com/	Nominal	1	Frequent	Yes	Label with Text	Thematic	Yes	Varying with vis position	Character	Around	both
35	Gran Turismo 7	Racing	https://www.granturismo.com/	Ordered	1	Frequent	Yes	Label with Numbers	Thematic	Yes	Varying with vis position	Character	Around	both
36	Gran Turismo 7	Racing	https://www.granturismo.com/	Spatial	1	Frequent	Yes	Sign	Salient	No	Varying with vis position	Location	Around	controlled
37	Gran Turismo 7	Racing	https://www.granturismo.com/	Nominal	2	Frequent	No	Map	Thematic	Yes	Fixed	Location	Around	controlled
38	Gran Turismo 7	Racing	https://www.granturismo.com/	Spatial; Quantitative	2	Rare	No	Ludophasmas	Thematic	No	Varying with vis position	Character	Integrated	both
39	Forza Horizon 5	Racing	https://www.xbox.com/en-us/games/forzahorizon5.html	Nominal	1	Frequent	No	Label with Text	Hidden	Yes	Varying with vis position	Location	Around	controlled
40	Forza Horizon 5	Racing	https://www.xbox.com/en-us/games/forzahorizon5.html	Quantitative; Spatial	2	Frequent	No	World annotation	Thematic	No	Fixed	Location	Integrated	controlled
41	Forza Horizon 5	Racing	https://www.xbox.com/en-us/games/forzahorizon5.html	Nominal	1	Frequent	Yes	Label with Text	Thematic	Yes	Varying with vis position	Character	Around	both
42	Forza Horizon 5	Racing	https://www.xbox.com/en-us/games/forzahorizon5.html	Ordered	1	Frequent	Yes	Label with Numbers	Thematic	Yes	Varying with vis position	Character	Around	both
43	Orcs Must Die! 3	RTS	https://www.xbox.com/en-us/games/orcsmustdie3.html	Quantitative	1	Frequent	Yes	Bar	Thematic	Yes	Varying with vis position	Character	Around	both
44	Orcs Must Die! 3	RTS	https://www.xbox.com/en-us/games/orcsmustdie3.html	Quantitative	1	Frequent	No	Label with Numbers	Salient	Yes	Varying with vis position	Character	Around	controlled
45	Orcs Must Die! 3	RTS	https://www.xbox.com/en-us/games/orcsmustdie3.html	Spatial	1	Frequent	Yes	Ludophasmas	Thematic	No	Varying with vis position	Location	Integrated	controlled

Figure A.2: Systematic Review Data - Page 2.

Figure A.3: Systematic Review Data - Page 3.

71	MechWarrior 5: Mercenaries	Simulation	https://mw5mercs.com	Spatial; Nominal	1	Frequent	No	Sign	Thematic	Yes	Varying with vis position	Character	Around	both
72	MechWarrior 5: Mercenaries	Simulation	https://mw5mercs.com	Nominal	2	Frequent	Yes	Sign	Thematic	Yes	Varying with vis position	Character	Around	both
73	Dirt 5	Simulation	https://dirtgame.com/d5ds	Nominal	1	Frequent	No	Label with Text	Thematic	Yes	Varying with vis position	Character	Around	both
74	Dirt 5	Simulation	https://dirtgame.com/d5ds	Ordered	1	Frequent	No	Label with Numbers	Thematic	Yes	Varying with vis position	Character	Around	both
75	Dirt 5	Simulation	https://dirtgame.com/d5ds	Spatial; Quantitative; Spatial	1	Frequent	Yes	Sign	Hidden	Yes	Varying with vis position	Character	Around	both
76	F1 2020	Simulation	https://formula1game.com/2020/	Quantitative; Spatial	2	Frequent	No	Word annotation	Thematic	No	Fixed Location	Integrated	controlled	controlled
77	F1 2020	Simulation	https://formula1game.com/2020/	Nominal	1	Frequent	No	Label with Text	Thematic	Yes	Varying with vis position	Character	Around	both
78	F1 2020	Simulation	https://formula1game.com/2020/	Ordered	1	Frequent	No	Label with Numbers	Thematic	Yes	Varying with vis position	Character	Around	both
79	NBA 2k21	Sports	https://nba.2k.com/nfl/FTR/	Nominal	1	Always Visible	No	Circle	Thematic	No	Varying with vis position	Character	Around	controlled
80	NBA 2k21	Sports	https://nba.2k.com/nfl/FTR/	Quantitative	1	Always Visible	No	Circular bar	Thematic	No	Varying with vis position	Character	Around	controlled
81	NBA 2k21	Sports	https://nba.2k.com/nfl/FTR/	Nominal	1	Always Visible	No	Label with Text	Hidden	No	Varying with vis position	Character	Around	controlled
82	NBA 2k21	Sports	https://nba.2k.com/nfl/FTR/	Quantitative	1	Frequent	No	Circular bar	Thematic	Yes	Varying with vis position	Character	Around	controlled
83	NBA 2k21	Sports	https://nba.2k.com/nfl/FTR/	Spatial	1	Frequent	No	Circular bar	Thematic	No	Fixed Location	Over	controlled	controlled
84	NBA 2k21	Sports	https://nba.2k.com/nfl/FTR/	Nominal	1	Frequent	No	Sign	Salient	Yes	Varying with vis position	Character	Around	both
85	Madden NFL 21	Sports	https://www.ea.com/games/madden-nfl/madden-nfl21	Nominal	1	Always Visible	No	Areas	Thematic	No	Varying with vis position	Character	Around	controlled
86	Madden NFL 21	Sports	https://www.ea.com/games/madden-nfl/madden-nfl21	Quantitative	1	Always Visible	Yes	Circular bar	Thematic	No	Varying with vis position	Character	Around	controlled
87	Madden NFL 21	Sports	https://www.ea.com/games/madden-nfl/madden-nfl21	Nominal	1	Frequent	No	Sign	Thematic	No	Varying with vis position	Character	Around	both
88	Madden NFL 21	Sports	https://www.ea.com/games/madden-nfl/madden-nfl21	Nominal	1	Frequent	Yes	Sign	Thematic	Yes	Varying with vis position	Character	Around	both
89	Phoenix Point	Strategy	https://phoenixpoint.info	Quantitative; Nominal	2	Frequent	No	Bar	Thematic	Yes	Varying with vis position	Character	Around	controlled
90	Phoenix Point	Strategy	https://phoenixpoint.info	Quantitative	1	Frequent	No	Bar	Salient	Yes	Varying with vis position	Character	Around	controlled
91	Phoenix Point	Strategy	https://phoenixpoint.info	Quantitative	2	Frequent	Yes	Areas	Thematic	No	Varying with vis position	Location	Over	controlled
92	Gears Tactics	Strategy	https://www.gearstactics.com	Quantitative; Nominal	2	Always Visible	No	Bar	Thematic	Yes	Varying with vis position	Character	Around	controlled
93	Gears Tactics	Strategy	https://www.gearstactics.com	Quantitative	1	Always Visible	No	Label with Numbers	Thematic	Yes	Varying with vis position	Character	Around	controlled
94	Gears Tactics	Strategy	https://www.gearstactics.com	Nominal	1	Frequent	Yes	Silhouettes	Salient	Yes	Varying with vis position	Character	Around	controlled
95	Gears Tactics	Strategy	https://www.gearstactics.com	Quantitative	1	Frequent	Yes	Areas	Thematic	No	Varying with vis position	Location	Over	controlled
96	Gears Tactics	Strategy	https://www.gearstactics.com	Spatial	1	Frequent	No	Sign	Thematic	Yes	Varying with vis position	Location	Around	controlled
97	Aliens: Fireteam Elite	TPS	https://AlienTeamElite.com/	Quantitative	1	Frequent	Yes	Bar	Salient	Yes	Varying with vis position	Character	Around	controlled

Figure A.4: Systematic Review Data - Page 4.

98	Allens: Fireteam Elite	TPS	https://www.alienstireteamelite.com/en/	Nominal	1	Frequent	No	Label with Text	Hidden	Yes	Varying with vis position	Character	Around	both
99	Allens: Fireteam Elite	TPS	https://www.alienstireteamelite.com/en/	Spatial	1	Moderate	No	Ludophasmas	Salient	No	Varying with vis position	Object	Integrated	controlled
100	Allens: Fireteam Elite	TPS	https://www.alienstireteamelite.com/en/	Spatial; Nominal	2	Frequent	No	Sign	Thematic	Yes	Varying with vis position	Object	Around	controlled
101	Allens: Fireteam Elite	TPS	https://www.alienstireteamelite.com/en/	Spatial	1	Frequent	No	Sign	Thematic	Yes	Varying with vis position	Location	Around	controlled
102	Allens: Fireteam Elite	TPS	https://www.alienstireteamelite.com/en/	Quantitative	1	Rare	No	Circular bar	Thematic	Yes	Varying with vis position	Character	Around	controlled
103	Allens: Fireteam Elite	TPS	https://www.alienstireteamelite.com/en/	Quantitative	1	Rare	No	Bar	Thematic	Yes	Varying with vis position	Character	Around	controlled
104	Returnal	TPS	https://housemarque.com/games/returnal/	Quantitative	1	Frequent	Yes	Bar	Salient	Yes	Varying with vis position	Character	Around	controlled
105	Returnal	TPS	https://housemarque.com/games/returnal/	Spatial	1	Rare	Yes	Sign	Thematic	Yes	Varying with vis position	Object	Around	controlled
106	Outriders	TPS	https://outriders.square-enix-games.com/en/	Spatial	1	Frequent	No	Sign	Thematic	No	Varying with vis position	Location	Around	controlled
107	Outriders	TPS	https://outriders.square-enix-games.com/en/	Spatial	1	Rare	No	Areas	Salient	No	Fixed	Location	Over	controlled
108	Outriders	TPS	https://outriders.square-enix-games.com/en/	Quantitative	1	Frequent	Yes	Bar	Thematic	Yes	Varying with vis position	Character	Around	controlled
109	Outriders	TPS	https://outriders.square-enix-games.com/en/	Spatial	1	Frequent	Yes	Sign	Thematic	Yes	Varying with vis position	Character	Around	both
110	King's Bounty II	Turn-Based Strategy	https://kingsbounty2.com/	Ordered	1	Frequent	No	Label with Numbers	Thematic	No	Fixed	Character	Around	both
111	King's Bounty II	Turn-Based Strategy	https://kingsbounty2.com/	Nominal	1	Frequent	No	Sign	Thematic	No	Varying with vis position	Character	Around	both
112	King's Bounty II	Turn-Based Strategy	https://kingsbounty2.com/	Spatial	1	Frequent	Yes	Heatmap	Thematic	No	Fixed	Location	Over	controlled
113	Sakura Wars	Turn-Based Strategy	https://games.sega.com/sakurawars/angle/index.html	Quantitative	1	Frequent	Yes	Bar	Salient	Yes	Varying with vis position	Character	Around	controlled
114	Sakura Wars	Turn-Based Strategy	https://games.sega.com/sakurawars/angle/index.html	Nominal	1	Frequent	Yes	Label with Text	Salient	Yes	Varying with vis position	Character	Around	controlled
115	Sakura Wars	Turn-Based Strategy	https://games.sega.com/sakurawars/angle/index.html	Quantitative	1	Frequent	Yes	Label with Numbers	Hidden	No	Varying with vis position	Character	Around	both
116	XCOM: Chimera Squad	Turn-Based Strategy	https://www.xcom.com/life/FRChimera-squad/	Nominal	2	Frequent	No	Areas	Thematic	No	Fixed	Location	Around	controlled
117	XCOM: Chimera Squad	Turn-Based Strategy	https://www.xcom.com/life/FRChimera-squad/	Quantitative; Nominal	2	Always Visible	Yes	Bar	Thematic	Yes	Varying with vis position	Character	Around	controlled
118	XCOM: Chimera Squad	Turn-Based Strategy	https://www.xcom.com/life/FRChimera-squad/	Nominal	1	Frequent	Yes	Silhouettes	Hidden	Yes	Varying with vis position	Character	Around	controlled
119	XCOM: Chimera Squad	Turn-Based Strategy	https://www.xcom.com/life/FRChimera-squad/	Ordered	1	Always Visible	No	Label with Numbers	Thematic	Yes	Fixed	Character	Around	both
120	XCOM: Chimera Squad	Turn-Based Strategy	https://www.xcom.com/life/FRChimera-squad/	Spatial	1	Frequent	Yes	Heatmap	Salient	No	Fixed	Location	Over	controlled
121	Worms Battlegrounds	Wargames	https://www.team17.com/games/worms-battlegrounds/	Nominal	1	Always Visible	No	Label with Text	Salient	Yes	Fixed	Character	Around	both
122	Worms Battlegrounds	Wargames	https://www.team17.com/games/worms-battlegrounds/	Quantitative	1	Always Visible	No	Label with Numbers	Salient	Yes	Fixed	Character	Around	both

Figure A.5: Systematic Review Data - Page 5.

123	Worms Battlegrounds	Wargames	https://www.team17.com/games/worms-battlegrounds/	Quantitative	1	Frequent	No	Bar	Salient	Yes	Varying with vis position	Character	Around	autonomous
124	Worms Battlegrounds	Wargames	https://www.team17.com/games/worms-battlegrounds/	Quantitative	1	Frequent	No	Label with Numbers	Salient	Yes	Varying with vis position	Character	Around	both
125	Toy Soldiers	Wargames	http://accelerategames.com/games/toy-soldiers.html	Quantitative	1	Frequent	Yes	Bar	Thematic	No	Varying with vis position	Character	Around	both
126	Toy Soldiers	Wargames	http://accelerategames.com/games/toy-soldiers.html	Quantitative	1	Frequent	No	Label with Numbers	Hidden	No	Varying with vis position	Character	Around	both
127	Toy Soldiers	Wargames	http://accelerategames.com/games/toy-soldiers.html	Nominal	1	Frequent	Yes	Color Overlapped	Thematic	Yes	Fixed	Object	Over	controlled
128	Toy Soldiers	Wargames	http://accelerategames.com/games/toy-soldiers.html	Spatial	1	Frequent	No	Areas	Hidden	No	Fixed	Location	Over	controlled
129	Sengoku	Wargames	https://www.paradoxinteractive.com/games/sengoku/1400	Nominal	1	Always Visible	No	Map	Thematic	No	Fixed	Object	Integrated	controlled
130	Sengoku	Wargames	https://www.paradoxinteractive.com/games/sengoku/1400	Spatial; Nominal	2	Frequent	No	World annotation	Salient	Yes	Fixed	Character	Integrated	both
131	Sengoku	Wargames	https://www.paradoxinteractive.com/games/sengoku/1400	Quantitative	1	Frequent	No	Bar	Thematic	Yes	Varying with vis position	Character	Around	both
132	Sengoku	Wargames	https://www.paradoxinteractive.com/games/sengoku/1400	Quantitative	1	Frequent	No	Label with Numbers	Thematic	Yes	Fixed	Character	Around	both
133	Sengoku	Wargames	https://www.paradoxinteractive.com/games/sengoku/1400	Nominal	1	Frequent	No	Sign	Thematic	Yes	Varying with vis position	Character	Around	both
134	Sengoku	Wargames	https://en.wikipedia.org/wiki/WWE_2K14	Nominal	1	Moderate	Yes	Circle	Salient	No	Varying with vis position	Character	Around	both
135	WWE 2k14	Wrestling	https://en.wikipedia.org/wiki/WWE_2K14	Quantitative	1	Frequent	No	Bar	Salient	Yes	Varying with vis position	Character	Around	controlled
136	WWE 2k14	Wrestling	https://en.wikipedia.org/wiki/WWE_2K14	Nominal	1	Rare	No	Pictograph	Salient	Yes	Varying with vis position	Character	Around	controlled
137	WWE 2k14	Wrestling	https://crappygames.mirahze.org/wiki/Fire_Pro_Wrestling_(2012)	Quantitative	1	Rare	No	Bar	Salient	Yes	Varying with vis position	Character	Around	controlled
138	Fire Pro Wrestling	Wrestling	https://crappygames.mirahze.org/wiki/Fire_Pro_Wrestling_(2012)	Quantitative	1	Rare	No	Bar	Salient	Yes	Varying with vis position	Character	Around	controlled
139	Fire Pro Wrestling	Wrestling	https://crappygames.mirahze.org/wiki/Fire_Pro_Wrestling_(2012)	Nominal	1	Frequent	No	Sign	Thematic	Yes	Varying with vis position	Character	Around	controlled
140	Fire Pro Wrestling	Wrestling	https://unknowntoys.com/subnautica/	Quantitative; Nominal	1	Moderate	No	Pie chart	Thematic	Yes	Varying with vis position	Character	Over	controlled
141	Subnautica: Below Zero	Adventure	https://unknowntoys.com/subnautica/	Quantitative; Nominal	2	Frequent	No	Sign	Thematic	Yes	Varying with vis position	Location	Around	controlled

Figure A.6: Systematic Review Data - Page 6.

Figure A.7: Systematic Review Data - Page 7.

A.3. Systematic review: Relationships between different dimensions



Figure A.8: Number of occurrences of visualizations moving autonomously, controlled by the player or depending on both factors depending on the type of data referent based on our systematic review.

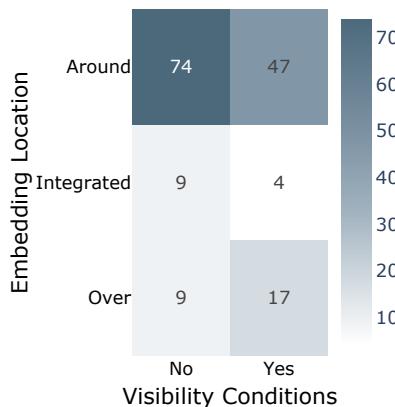


Figure A.9: Number of occurrences of visualizations embedded in the referent in different locations per each visibility condition (either on focus or not), based on our systematic review.

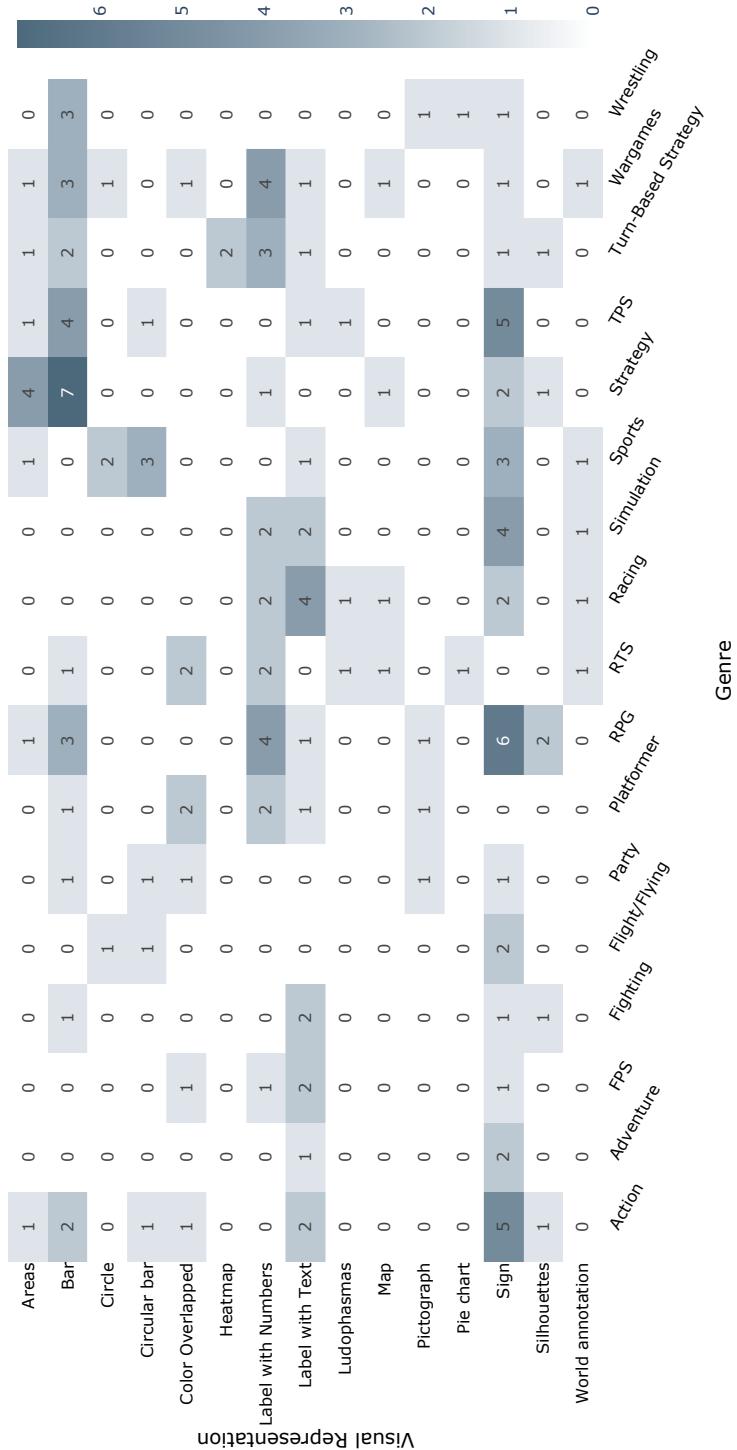


Figure A.10: Number of occurrences of visualizations represented by a specific type of visual representation in different game genres, based on our systematic review.

A.4. Visualization design sketches



(a) Design 1 - Bar chart placed above the head of the character. Using color+length encoding or length only.



(b) Design 2 - Donut chart placed below the feet of the character. Using color+length encoding or length only.



(c) Design 3 - Bar chart placed on the side of the character. Using color+length encoding or length only.



(d) Design 4 - Sign placed above the head of the character. Using color only encoding.



(e) Design 5 - Rounded area placed below the feet of the character. Using color+area encoding or color only.



(f) Design 6 - Heart-shape pictorial fraction chart placed above the head of the character. Using color+length encoding or length only.

Figure A.11: Part 1 - Sketches of Health level visualization to be embedded around the character (the robot).



(a) Design 7 - Arrow-shape pictorial fraction chart placed above the head of the character. Using color+length encoding or length only.



(b) Design 8 - Segmented bar chart placed above the head of the character. Using color+length encoding.



(c) Design 9 - Pictograph chart placed above the head of the character. Using heart-shape icons number encoding.



(d) Design 10 - Label with text placed above the head of the character. Using color+number encoding or number only.



(e) Design 11 - Pie chart placed above the head of the character. Using color+angle encoding or angle only.



(f) Design 12 - Bar chart placed above the head of the character without background. Using color+length encoding or length only.

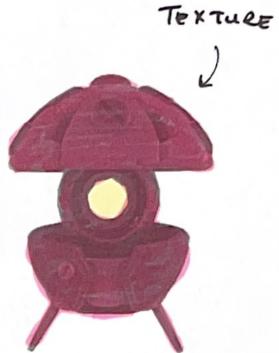
Figure A.12: Part 2 - Sketches of Health level visualization to be embedded around the character (the robot).



(a) Design 13 - Bar chart placed on the top part of the character body. Using color+length encoding or length only.



(b) Design 14 - Colored texture of the eye of the character. Using color only encoding.



(c) Design 15 - Colored texture of the whole body of the character. Using color+length encoding or length only.



(d) Design 16 - Pie chart embedded in the eye of the character. Using angle+color encoding or angle only.



(e) Design 17 - Colored elements on the top part of the body of the character. Using color+number encoding.



(f) Design 18 - Donut chart embedded in the eye of the character. Using color+length encoding or length only.



(g) Design 19 - Silhouette overlapping the body of the character. Using color+length encoding or length only.



(h) Design 20 - Bar chart overlapping the body of the character. Using color+length encoding or length only.



(i) Design 21 - Donut chart overlapping the body of the character. Using color+length encoding or length only.

Figure A.13: Sketches of Health level visualization to be integrated in the character's (the robot) design or overlapping the character body.

A.5. Game mechanics implementation scripts

Listing A.1: ObjectiveKillEvilRobots.cs

```
1  using UnityEngine;
2  using UnityEngine;
3
4  namespace Unity.FPS.Gameplay
5  {
6      public class ObjectiveKillEvilRobots : Objective
7      {
8
9          [SerializeField]
10         [Tooltip("Total number of evil robot to eliminate")]
11         private int evilRobotNumber = 8;
12
13         private int evilRobotEliminated = 0;
14
15         [SerializeField]
16         [Tooltip("All the enemies in the scenario")]
17         private GameObject[] robots;
18
19
20         // Affiliation Static values
21         private static int GOOD = 0;
22         private static int EVIL = 1;
23
24         private static int[] healthValues = { 18, 32, 43, 58, 72, 83
25             };
26         private int currentIndex = 0;
27         private int robotPerValue = 4;
28         private float healthThreeshold = 66f;
29
30         // Use this for initialization
31         protected override void Start()
32         {
33
34             Title = "Eliminate " + evilRobotNumber + " evil robots";
35             Description = "Their health must be higher than 66%";
36
37             base.Start();
38
39             EventManager.AddListener<EnemyKillEvent>(OnKill);
```

```

40         // initialize the robots
41         assignRobotsAffiliation();
42     }
43
44     void assignRobotsAffiliation()
45     {
46         // shuffle robot list
47         robots = Shuffle(robots);
48
49         // for each health value instantiate #robotPerValue robots
50         // assigning affiliation and health
51         foreach (int healthValue in healthValues)
52         {
53             for(int i = currentIndex; i < currentIndex +
54                 robotPerValue; i++)
55             {
56                 generateHealth(i, healthValue);
57             }
58             currentIndex += robotPerValue;
59         }
60     }
61
62     void generateHealth(int i, int healthValue)
63     {
64         Actor actor = robots[i].GetComponent<Actor>();
65         Health health = robots[i].GetComponent<Health>();
66         health.CurrentHealth = healthValue;
67
68         if (healthValue > healthThreeshold)
69             actor.Affiliation = EVIL;
70         else
71             actor.Affiliation = GOOD;
72     }
73
74     public GameObject[] Shuffle(GameObject[] objectList)
75     {
76         GameObject tempGO;
77
78         for (int i = 0; i < objectList.Length; i++)
79         {
80             int rnd = Random.Range(0, objectList.Length);
81             tempGO = objectList[rnd];
82             objectList[rnd] = objectList[i];

```

```
83             objectList[i] = tempGO;
84         }
85
86         return objectList;
87     }
88
89     void OnKill(EnemyKillEvent evt)
90     {
91         GameObject enemy = evt.Enemy.gameObject;
92         int affiliation = enemy.GetComponent<Actor>().Affiliation
93             ;
94
95         if (affiliation == EVIL)
96         {
97             // decrease number of robot to eliminate
98             evilRobotEliminated++;
99
100            // log event
101            KillEvilRobotEvent killEvent = new KillEvilRobotEvent
102                ();
103            EventManager.Broadcast(killEvent);
104
105            //check Winnind Conditions
106            checkWinningCondition();
107        }
108        else if(affiliation == GOOD)
109        {
110            // log event
111            KillGoodRobotEvent killEvent = new KillGoodRobotEvent
112                ();
113            EventManager.Broadcast(killEvent);
114        }
115
116        void checkWinningCondition()
117        {
118            if (evilRobotEliminated < evilRobotNumber)
119                UpdateObjective(string.Empty, evilRobotEliminated + "
120                    /" + evilRobotNumber, evilRobotNumber -
121                    evilRobotEliminated + " evil robots left");
122        }
123
124        else
125            CompleteObjective(string.Empty, string.Empty, "
126                Objective complete : " + Title);
```

```
121
122     }
123
124     void OnDestroy()
125     {
126         EventManager.RemoveListener<EnemyKillEvent>(OnKill);
127     }
128 }
129 }
```

Listing A.2: ObjectiveSaveGoodRobots.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using Unity.FPS.Game;
5
6 namespace Unity.FPS.Gameplay
7 {
8     public class ObjectiveSaveGoodRobots : Objective
9     {
10         [Tooltip("Number of enemy that you could kill by mistake")]
11         private int mistakesAllowed = 3;
12
13         private int mistakesDone = 0;
14
15         [SerializeField]
16         [Tooltip("Player health reference")]
17         private Health player;
18
19         private List<GameObject> damagedByMistake = new List<
20             GameObject>();
21
22         // Affiliation Static values
23         private static int GOOD = 0;
24         // Affiliation Static values
25         private static int EVIL = 1;
26
27         // Start is called before the first frame update
28         protected override void Start()
29         {
30             Title = "Ignore the Good Robots";
31             Description = "Don't damage more than " + (
32                 mistakesAllowed - 1) + " Good robots";
33         }
34     }
35 }
```

```
31
32     // Always a Secondary Objective, paired with
33     // ObjectiveKillEvilRobots
34     IsOptional = true;
35
36     EventManager.AddListener<DamageEvent>(OnDamage);
37
38     base.Start();
39 }
40
41 void OnDamage(DamageEvent evt)
42 {
43
44     if (evt.enemyAffiliation == GOOD)
45     {
46
47         if (!damagedByMistake.Contains(evt.EnemyDamaged))
48         {
49
50             damagedByMistake.Add(evt.EnemyDamaged);
51             mistakesDone++;
52
53             UpdateObjective(string.Empty, mistakesDone + "/" +
54                             mistakesAllowed, "You can damage a good
55                             robot " + (mistakesAllowed - mistakesDone) + " "
56                             "time left");
57
58             if (mistakesDone == mistakesAllowed)
59                 player.Kill();
60
61
62         else if (evt.enemyAffiliation == EVIL)
63         {
64
65             HitEvilRobotEvent hit = new HitEvilRobotEvent();
66             EventManager.Broadcast(hit);
67
68         }
69
70     void OnDestroy()
71     {
72
73         EventManager.RemoveListener<DamageEvent>(OnDamage);
```

```
71         }
72     }
73 }
```

Listing A.3: EnemyController.cs

```
1  using System.Collections.Generic;
2  using Unity.FPS.Game;
3  using UnityEngine;
4  using UnityEngine.AI;
5  using UnityEngine.Events;
6
7  namespace Unity.FPS.AI
8 {
9      [RequireComponent(typeof(Health), typeof(Actor), typeof(
10         NavMeshAgent))]
11      public class EnemyController : MonoBehaviour
12      {
13          [...]
14
15          [SerializeField]
16          [Tooltip("The enemy attack only when attacked the first time
17             or not")]
18          private bool attacksOnlyOnDamage = false;
19
20          protected virtual void Update()
21          {
22              EnsureIsWithinLevelBounds();
23
24              if (!attacksOnlyOnDamage)
25                  DetectionModule.HandleTargetDetection(m_Actor,
26                                              m_SelfColliders);
27
28              Color currentColor = OnHitBodyGradient.Evaluate((Time.
29                  time - m_LastTimeDamaged) / FlashOnHitDuration);
30              m_BodyFlashMaterialPropertyBlock.SetColor("_EmissionColor
31                  ", currentColor);
32              foreach (var data in m_BodyRenderers)
33              {
34                  data.Renderer SetPropertyBlock(
35                      m_BodyFlashMaterialPropertyBlock, data.
36                      MaterialIndex);
37              }
38          }
39
40 }
```

```
32
33         m_WasDamagedThisFrame = false;
34     }
35
36     void OnDamaged(float damage, GameObject damageSource)
37     {
38         // test if the damage source is the player
39         if (damageSource && !damageSource.GetComponent<
40             EnemyController>())
41         {
42             // pursue the player
43             DetectionModule.OnDamaged(damageSource);
44
45             onDamaged?.Invoke();
46             m_LastTimeDamaged = Time.time;
47
48             // play the damage tick sound
49             if (DamageTick && !m_WasDamagedThisFrame)
50                 AudioUtility.CreateSFX(DamageTick, transform.
51                     position, AudioUtility.AudioGroups.DamageTick,
52                     0f);
53
54             m_WasDamagedThisFrame = true;
55
56             // Send DamageEvent
57             DamageEvent evt = new DamageEvent();
58             evt.EnemyDamaged = this.gameObject;
59             evt.enemyAffiliation = m_Actor.Affiliation;
60             EventManager.Broadcast(evt);
61
62             // Handles attacking only on first damage
63             if (attacksOnlyOnDamage)
64                 DetectionModule.HandleTargetDetection(m_Actor,
65                     m_SelfColliders);
66         }
67     }
68
69     public Vector3 RandomNavDestination(float radius)
70     {
71         Vector3 randomDirection = transform.position + Random.
72             insideUnitSphere * radius;
73         NavMeshHit hit;
74
75         if (NavMesh.SamplePosition(randomDirection, out hit,
76             0.1f, 1)
```

```

        radius, NavMesh.AllAreas))
71     {
72         finalPosition = hit.position;
73     }
74     return finalPosition;
75 }
76 }
77 }
```

Listing A.4: EnemyMobile.cs

```

1 using Unity.FPS.Game;
2 using UnityEngine;
3
4 namespace Unity.FPS.AI
5 {
6     [RequireComponent(typeof(EnemyController))]
7     public class EnemyMobile : MonoBehaviour
8     {
9         public enum AIState
10        {
11            Patrol,
12            Follow,
13            Attack,
14        }
15
16        public bool allowFreeNavigation = false;
17
18        [Tooltip("Radius of free navigation at each step when in idle
19          ")]
20        public float navigationRadius = 5f;
21        [Tooltip("Fraction of the navigation range at which the enemy
22           will change destination point")]
23        public float sensibilityOfDistance = 0.5f;
24
25        float navigationTime = 0f;
26        [SerializeField]
27        float totalNavigationTime = 3f;
28
29        [...]
30
31        void Update()
```

```
32         {
33             UpdateCurrentAiState();
34
35             [...]
36         }
37
38     void UpdateCurrentAiState()
39     {
40         // Handle logic
41         switch (AiState)
42         {
43             case AIState.Patrol:
44                 m_EnemyController.UpdatePathDestination();
45
46                 // Free navigation robot
47                 if(allowFreeNavigation)
48                 {
49                     navigationTime += Time.deltaTime;
50
51                     if(navigationTime >= totalNavigationTime)
52                     {
53                         navigationTime = 0f;
54                         m_EnemyController.SetNavDestination(
55                             m_EnemyController.RandomNavDestination(
56                             navigationRadius));
57                     }
58                 }
59             else
60                 m_EnemyController.SetNavDestination(
61                     m_EnemyController.GetDestinationOnPath());
62                 break;
63             case AIState.Follow:
64                 m_EnemyController.SetNavDestination(
65                     m_EnemyController.KnownDetectedTarget.
66                     transform.position);
67                 m_EnemyController.OrientTowards(m_EnemyController.
68                     KnownDetectedTarget.transform.position);
69                 m_EnemyController.OrientWeaponsTowards(
70                     m_EnemyController.KnownDetectedTarget.
71                     transform.position);
72                 break;
73             case AIState.Attack:
74                 if (Vector3.Distance(m_EnemyController.
75                     KnownDetectedTarget.transform.position,
```

```

67                         m_EnemyController.DetectionModule.
68                             DetectionSourcePoint.position)
69             >= (AttackStopDistanceRatio *
70                 m_EnemyController.DetectionModule.
71                 AttackRange))
72         {
73             m_EnemyController.SetNavDestination(
74                 m_EnemyController.KnownDetectedTarget.
75                     transform.position);
76         }
77     else
78     {
79         m_EnemyController.SetNavDestination(transform
80             .position);
81     }
82 }
83 }
```

Listing A.5: ObjectiveResourcesPickUp.cs

```

1  using System.Collections.Generic;
2  using Unity.FPS.Game;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  namespace Unity.FPS.Gameplay
7  {
8      public class ObjectiveResourcesPickUp : Objective
9      {
10          [Tooltip("Total number of resources to pickup to complete the
11              objective")]
12          public int numberToPickUp;
13
14          [Tooltip("Max number of pickup actions allowed")]
15          public int maxPickup;
```

```
16     [Tooltip("Image for the pickup actions")]
17     public Image pickupImage;
18
19     [Tooltip("The transform for the resources allocation
20         positions")]
21     public Transform[] resourcePositions;
22
23     [Tooltip("The UI element for the pickup command")]
24     public GameObject pickupCommand;
25
26     [Tooltip("The text element for bolt counter")]
27     public Text boltCounter;
28
29     [Tooltip("Image for the bolt counter")]
30     public Image boltImage;
31
32     [Tooltip("The text element for gear counter")]
33     public Text gearCounter;
34
35     [Tooltip("Image for the gear counter")]
36     public Image gearImage;
37
38     [Tooltip("The text element for battery counter")]
39     public Text batteryCounter;
40
41     [Tooltip("Image for the battery counter")]
42     public Image batteryImage;
43
44     [Tooltip("The text element for pickup counter")]
45     public Text pickupCounter;
46
47     [Tooltip("Audio source")]
48     public AudioSource AudioSource;
49
50     [Tooltip("Sound played when there are not more pickupActions
51         available")]
52     public AudioClip noMorePickupSound;
53
54     public GameObject bolt;
55     public GameObject gear;
56     public GameObject power;
57
58     // current number of pickup actions done
```

```
58     public int currentPickup = 0;
59     // List of items to pickup
60     private List<ResourcePickup.Type> toPickup = new List<
61         ResourcePickup.Type>();
62     // list of items picked up
63     private List<ResourcePickup.Type> pickedUp = new List<
64         ResourcePickup.Type>();
65
66     private int countGear;
67     private int countBolt;
68     private int countBattery;
69
70     private int pickedBolt = 0;
71     private int pickedGear = 0;
72     private int pickedBattery = 0;
73
74     bool reachedMaxPickup = false;
75
76     protected override void Start()
77     {
78         base.Start();
79
80         EventManager.AddListener<PickupResourceEvent>(
81             OnPickupEvent);
82
83         createPickupList();
84         displayResources();
85
86         pickupCounter.text = maxPickup.ToString();
87     }
88
89     void displayResources()
90     {
91         int i = 0;
92         //make sure the elements to complete the objective are
93         //all there always
94         foreach(ResourcePickup.Type resource in toPickup)
95         {
96             if ((int)resource == (int)ResourcePickup.Type.Bolt)
97             {
98                 GameObject newResource = Instantiate(bolt,
```

```
        resourcePositions[i].position, Quaternion.
        identity);
    newResource.transform.parent = resourcePositions[
        i];
    newResource.transform.parent.GetComponent<
        ResourcePickup>().type = ResourcePickup.Type.
        Bolt;
}
else if ((int)resource == (int)ResourcePickup.Type.
        Gear)
{
    GameObject newResource = Instantiate(gear,
        resourcePositions[i].position, Quaternion.
        identity);
    newResource.transform.parent = resourcePositions[
        i];
    newResource.transform.parent.GetComponent<
        ResourcePickup>().type = ResourcePickup.Type.
        Gear;
}
else if ((int)resource == (int)ResourcePickup.Type.
        Battery)
{
    GameObject newResource = Instantiate(power,
        resourcePositions[i].position, Quaternion.
        identity);
    newResource.transform.parent = resourcePositions[
        i];
    newResource.transform.parent.GetComponent<
        ResourcePickup>().type = ResourcePickup.Type.
        Battery;
}
i++;
}
// display the remaining resources in the scene
randomly
for (i = numberToPickUp; i < resourcePositions.Length; i
++)
{
    ResourcePickup.Type resource = (ResourcePickup.Type)
        Random.Range(0, 3);

    if ((int)resource == (int)ResourcePickup.Type.Bolt)
```

```
122 {
123     GameObject newResource = Instantiate(bolt,
124         resourcePositions[i].position, Quaternion.
125             identity);
126     newResource.transform.parent = resourcePositions[
127         i];
128     newResource.transform.parent.GetComponent<
129         ResourcePickup>().type = ResourcePickup.Type.
130         Bolt;
131 }
132     else if ((int)resource == (int)ResourcePickup.Type.
133         Gear)
134 {
135     GameObject newResource = Instantiate(gear,
136         resourcePositions[i].position, Quaternion.
137             identity);
138     newResource.transform.parent = resourcePositions[
139         i];
140     newResource.transform.parent.GetComponent<
141         ResourcePickup>().type = ResourcePickup.Type.
142         Gear;
143 }
144     else if ((int)resource == (int)ResourcePickup.Type.
145         Battery)
146 {
147     GameObject newResource = Instantiate(power,
148         resourcePositions[i].position, Quaternion.
149             identity);
150     newResource.transform.parent = resourcePositions[
151         i];
152     newResource.transform.parent.GetComponent<
153         ResourcePickup>().type = ResourcePickup.Type.
154         Battery;
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 void createPickupList()
164 {
165     // fill the list of items to pick up randomly
166     for (int i = 0; i < numberToPickUp; i++)
167     {
168         ResourcePickup.Type resource = (ResourcePickup.Type)
```

```
        Random.Range(0, 3);
149    toPickup.Add(resource);

150
151    if ((int)resource == (int)ResourcePickup.Type.Bolt)
152    {
153        countBolt++;
154    }
155    else if ((int)resource == (int)ResourcePickup.Type.
156        Gear)
157    {
158        countGear++;
159    }
160    else if ((int)resource == (int)ResourcePickup.Type.
161        Battery)
162    {
163        countBattery++;
164    }
165
166    boltCounter.text = "0/" + countBolt;
167    gearCounter.text = "0/" + countGear;
168    batteryCounter.text = "0/" + countBattery;
169
170    void OnPickupEvent(PickupResourceEvent evt)
171    {
172
173        if (IsCompleted)
174            return;
175
176        // if there are no more pickup actions available
177        if (reachedMaxPickup)
178        {
179            AudioSource.PlayOneShot(noMorePickupSound, 1F);
180        }
181        else
182        {
183            // handle pickup
184            ResourcePickup picked = (ResourcePickup)evt.Pickup.
185                GetComponent<ResourcePickup>();
186
187            pickedUp.Add(picked.type);
188            updatePickUpCounting(picked.type);
```

```
189         // update counters
190         boltCounter.text = pickedBolt + "/" + countBolt;
191         gearCounter.text = pickedGear + "/" + countGear;
192         batteryCounter.text = pickedBattery + "/" +
193                         countBattery;
194
195         pickupCommand.SetActive(false);
196         Destroy(evt.Pickup);
197         currentPickup++;
198         pickupCounter.text = (maxPickup - currentPickup).
199                         ToString();
200
201         if (currentPickup == maxPickup)
202         {
203             reachedMaxPickup = true;
204             pickupImage.color = Color.red;
205
206             // If objective not reached already: player dies
207             if (!allPickedUp())
208             {
209                 PlayerDeathEvent deathEvent = new
210                     PlayerDeathEvent();
211                 EventManager.Broadcast(deathEvent);
212             }
213         }
214
215
216     void updatePickUpCounting(ResourcePickup.Type picked)
217     {
218         if ((int)picked == (int)ResourcePickup.Type.Bolt)
219         {
220             pickedBolt++;
221             if (pickedBolt == countBolt)
222                 boltImage.color = Color.green;
223         }
224         else if ((int)picked == (int)ResourcePickup.Type.Gear)
225         {
226             pickedGear++;
227             if (pickedGear == countGear)
228                 gearImage.color = Color.green;
229         }
230     }
231 }
```

```

230             else if ((int)picked == (int)ResourcePickup.Type.Battery)
231             {
232                 pickedBattery++;
233                 if (pickedBattery == countBattery)
234                     batteryImage.color = Color.green;
235             }
236         }
237
238     public bool allPickedUp()
239     {
240         return pickedBolt >= countBolt && pickedGear >= countGear
241             && pickedBattery >= countBattery;
242     }
243
244     void OnDestroy()
245     {
246         EventManager.RemoveListener<PickupResourceEvent>(
247             OnPickupEvent);
248     }
249 }
```

Listing A.6: EnemyGenerationPoint.cs

```

1 using UnityEngine;
2 using Unity.FPS.Game;
3
4 public class EnemyGenerationPoint : MonoBehaviour
5 {
6     [Tooltip("Enemy prefab to instantiate")]
7     public GameObject enemyPrefab;
8     [Tooltip("time interval between one enemy generation to the next
9         one")]
10    public float generationInterval;
11
12    float idleTime = 0f;
13    bool isIdle = false;
14    GameObject generatedEnemy;
15
16    // Start is called before the first frame update
17    void Start()
18    {
19        generatedEnemy = Instantiate(enemyPrefab, transform.position,
20            Quaternion.identity);
```

```
19         EventManager.AddListener<EnemyKillEvent>(OnEnemyKilled);  
20     }  
21  
22     private void Update()  
23     {  
24         if(isIdle) {  
25             idleTime += Time.deltaTime;  
26  
27             if(idleTime >= generationInterval)  
28             {  
29                 isIdle = false;  
30                 idleTime = 0f;  
31                 Instantiate(enemyPrefab, transform.position,  
32                             Quaternion.identity);  
33             }  
34         }  
35     }  
36     void OnEnemyKilled(EnemyKillEvent evt)  
37     {  
38         if(evt.Enemy.gameObject == generatedEnemy)  
39             isIdle = true;  
40     }  
41  
42     private void OnDestroy()  
43     {  
44         EventManager.RemoveListener<EnemyKillEvent>(OnEnemyKilled);  
45     }  
46  
47 }
```

A.6. Visualization Implementation: Colored texture Shader

Listing A.7: HealthBar.shader

```

1 Shader "Custom/HealthBar"
2 {
3     Properties
4     {
5         _Color ("Color", Color) = (1,1,1,1)
6         _HighlightColor ("Highlight Color", Color) = (1,1,1,1)
7         _MainTex ("Albedo (RGB)", 2D) = "white" {}
8         _Glossiness ("Smoothness", Range(0,1)) = 0.5
9         _Metallic ("Metallic", Range(0,1)) = 0.0
10        _HighPerc ("Height percentage", Range(0, 1)) = 0.0
11        _LocalMin ("Local Height Minimum", Float) = 0.0
12        _LocalMax ("Local Height Maximum", Float) = 1.0
13    }
14    SubShader
15    {
16        Tags{ "DisableBatching" = "true" }
17
18        Tags { "RenderType"="Opaque" }
19        LOD 200
20
21        CGPROGRAM
22        // Physically based Standard lighting model, and enable
23        // shadows on all light types
24        #pragma surface surf Standard fullforwardshadows vertex:vert
25
26        // Use shader model 3.0 target, to get nicer looking lighting
27        #pragma target 3.0
28
29        sampler2D _MainTex;
30
31        struct Input
32        {
33            float2 uv_MainTex;
34            float3 localPos;
35        };
36
37        half _Glossiness;
38        half _Metallic;

```

```
38     fixed4 _Color;
39     fixed _HighPerc;
40     fixed _LocalMax;
41     fixed _LocalMin;
42     fixed4 _HighlightColor;
43
44     // #pragma instancing_options assumeuniformscaling
45     UNITY_INSTANCING_BUFFER_START(Props)
46
47     UNITY_INSTANCING_BUFFER_END(Props)
48
49     void vert (inout appdata_full v, out Input o) {
50         UNITY_INITIALIZE_OUTPUT(Input,o);
51         o.localPos = v.vertex.xyz;
52     }
53
54     void surf (in Input IN, inout SurfaceOutputStandard o)
55     {
56         // Albedo comes from a texture tinted by color
57
58         fixed4 c = tex2D (_MainTex, IN.uv_MainTex);
59         o.Albedo = c.rgb;
60         if((IN.localPos.y - _LocalMin)/(_LocalMax - _LocalMin) >
61             _HighPerc)
62             o.Albedo *= _Color.rgb;
63         else
64             o.Albedo *= _HighlightColor.rgb;
65         // Metallic and smoothness come from slider variables
66         o.Metallic = _Metallic;
67         o.Smoothness = _Glossiness;
68         o.Alpha = c.a;
69     }
70     ENDCG
71 }
72 Fallback "Diffuse"
```

A.7. Pre-Questionnaire

1. Which gender do you identify with?

- woman
 - men
 - non binary / gender diverse
 - prefer not to tell
 - other:

2. What is your age?

3. How proficient or skilled do you believe you are at playing video games?

A horizontal timeline diagram consisting of six vertical bars of equal height, evenly spaced along a horizontal axis. The first bar on the left is labeled '← novice' at its top left corner. The last bar on the right is labeled 'expert →' at its top right corner.

4. How frequently do you play video games?

- daily
 - weekly
 - once or twice a month
 - once or twice in six months
 - once or twice in a year
 - once every few years

5. How frequently do you play First Person Shooter (FPS) games?

- daily
 - weekly
 - once or twice a month
 - once or twice in six month

- once or twice in a year
 once every few years

6. How proficient or skilled do you believe you are at playing video games on PC with Mouse + Keyboard as Input Devices?

← novice →

7. How frequently do you play video games on PC?

- daily
 weekly
 once or twice a month
 once or twice in six month
 once or twice in a year
 once every few years

8. What types of video games do you usually play? (check all that apply)

- FPS (First Person Shooter)
 Platformers
 Action/Adventure
 RPG (Role Playing Games)
 TPS (third Person Shooter)
 Sport games
 Puzzle games

A.8. In-game Questionnaire

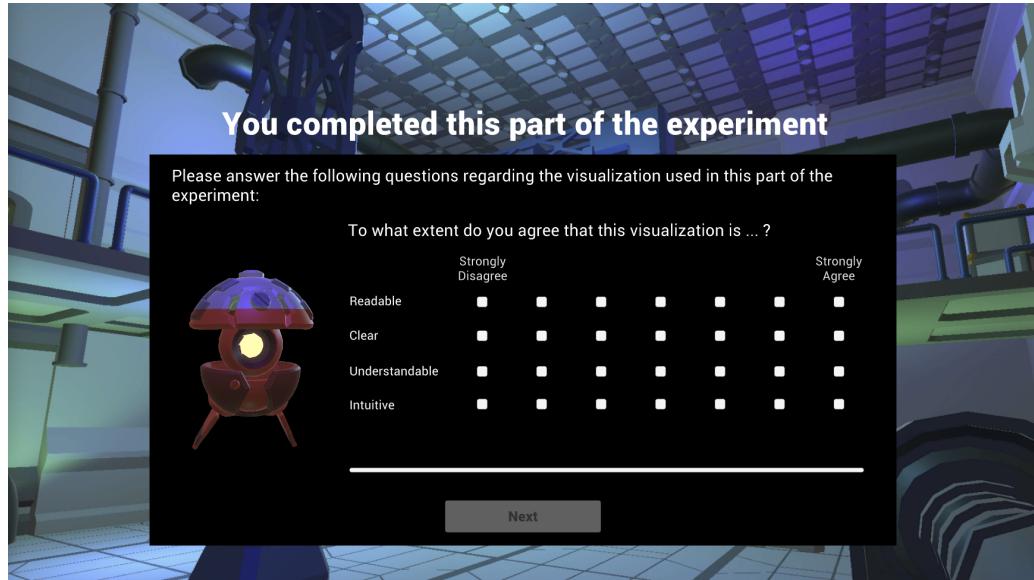


Figure A.14: In game questionnaire proposed inside RobotLife: Likert Scale of 4 items about the visualization's readability.

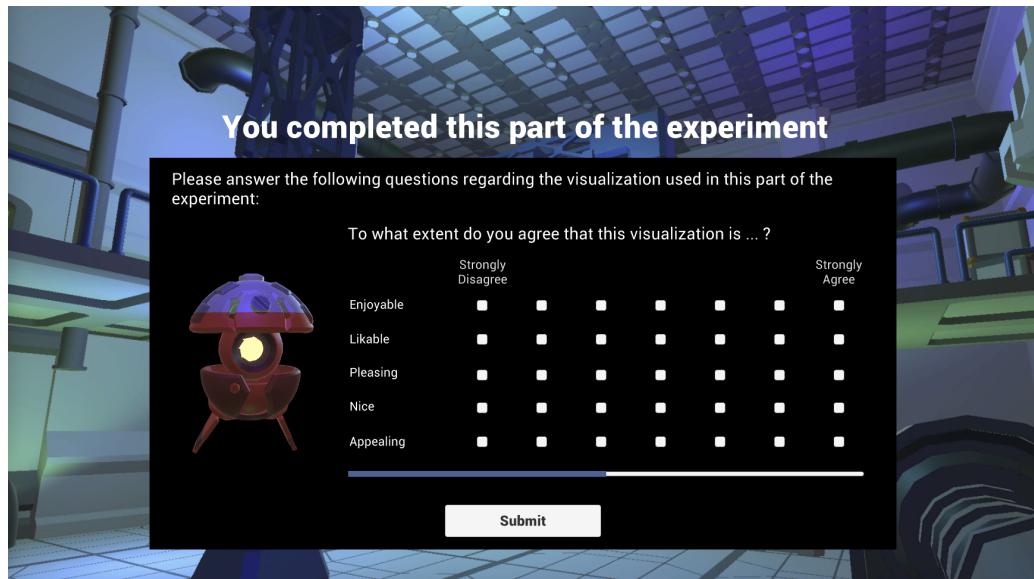


Figure A.15: In game questionnaire proposed inside RobotLife: Likert Scale of 5 item about the visualization's aesthetic. The scale used is BeauVis: A Validated Scale for Measuring the Aesthetic Pleasure of Visual Representations by He et al. [22].

A.9. Post-Questionnaire

- 1. Please rank the 3 visualization according to how easy they were to read while in motion**

First Choice (easier):

Second Choice:

Third Choice (more difficult):

- 2. Please rank the 3 visualizations according to how well they fit into the aesthetics of the game**

First Choice (fits better):

Second Choice:

Third Choice (fits less):

- 3. Please rank the 3 visualizations according to how well they supported you completing your mission**

First Choice (more supportive):

Second Choice:

Third Choice (less supportive):

- 4. Please rank the 3 visualizations according to how supportive you found them for a positive overall game experience (the game was fun, engaging, immersion, ...)**

First Choice (more supportive):

Second Choice:

Third Choice (less supportive):

A.10. Pilot Study results

Figure A.16: Data about the game performances. A logging system is implemented inside RobotLife and recorded all the salient actions of the player during the experiment. The log files produced have been analyzed to obtain the aggregated data reported in this table.

Date and Time	Participant ID	Which gender do you identify with?	What is your age?	How proficient or skilled do you believe you are at playing video games?	How frequently do you play video games?	How many hours a week do you play First Person Shooter (FPS) games?	How proficient or skilled do you believe you are at playing video games on PC with Mouse + Keyboard as Input Devices?	How frequently do you play video games on PC?	What types of video games do you usually play? (check all that apply)
19/07/2022 10:29:06	1	Woman	25	4	weekly	once every few years	3	once or twice a month	Platformers, RPG (Role Playing Games), Puzzle games
19/07/2022 14:38:50	3	Men	25	5	weekly	once every few years	5	weekly	Platformers, RPG (Role Playing Games)
21/07/2022 13:45:22	5	Men	26	5	weekly	once or twice a month	4	weekly	FPS (First Person Shooter), Platformers, Action/Adventure, RPG (Role Playing Games), TPS (Third Person Shooter), Puzzle games
21/07/2022 16:56:24	2	Woman	26	1	once every few years	once every few years	1	once every few years	Puzzle games
26/07/2022 12:56:39	4	Men	23	5	weekly	once or twice a month	4	once or twice a month	FPS (First Person Shooter), Platformers, Action/Adventure, RPG (Role Playing Games), TPS (Third Person Shooter), Sport games
26/07/2022 15:05:57	6	Woman	32	5	daily	once every few years	4	once every few years	TPS (Third Person Shooter)
28/07/2022 12:53:25	7	Woman	24	2	weekly	once every few years	2	weekly	RPG (Role Playing Games)
28/07/2022 15:30:35	8	Woman	24	4	daily	once every few years	4	once or twice a month	Action/Adventure, RPG (Role Playing Games), Sport games, Puzzle games
01/08/2022 13:35:02	9	Men	26	5	once or twice a month	once or twice in a year	5	once or twice a month	RPG (Role Playing Games)
03/08/2022 11:04:11	10	Men	25	3	once or twice a month	once or twice in a year	3	once or twice in a year	FPS (First Person Shooter), Action/Adventure
03/08/2022 13:04:13	11	Men	27	3	once or twice a month	once or twice a month	2	once or twice in six month	FPS (First Person Shooter), RPG (Role Playing Games)
03/08/2022 15:05:05	12	Men	22	4	weekly	once or twice in a year	3	once or twice in six month	Platformers, Action/Adventure, RPG (Role Playing Games)

Figure A.17: Data collected from the Pre-Questionnaire conducted on Google Forms.

Participant ID	Condition	Readability Likert Scale			Aesthetic likert scale					
		Readable	Clear	Understandable	Intuitive	Enjoyable	Likable	Pleasing	Nice	Appealing
1	A	7	7	7	7	6	6	6	6	6
	B	3	3	5	3	2	2	2	2	4
	C	5	5	4	5	5	5	5	5	5
2	A	5	6	2	3	4	3	3	3	2
	C	3	4	2	5	5	5	5	5	5
	B	2	2	2	2	6	6	6	6	6
3	B	6	5	7	6	6	6	6	6	6
	A	7	7	6	7	4	5	4	5	4
	C	5	5	6	6	4	4	4	5	4
4	B	6	7	5	6	6	6	6	7	6
	C	7	6	6	5	6	6	5	6	5
	A	7	7	7	7	6	6	5	6	6
5	C	6	6	6	6	6	6	6	6	6
	A	7	6	6	7	7	6	5	5	3
	B	4	6	7	7	7	7	7	7	7
6	C	5	4	5	4	5	5	5	4	4
	B	5	5	5	5	5	5	4	4	4
	A	6	6	6	5	6	5	5	5	5
7	A	7	4	7	6	5	5	5	6	4
	B	3	3	3	5	7	6	3	3	6
	C	5	2	2	4	1	4	3	3	6
8	A	5	4	6	7	7	7	3	4	4
	C	2	6	7	7	7	7	2	4	7
	B	7	7	7	7	6	7	7	7	7
9	B	6	5	6	5	6	6	5	5	5
	A	6	5	7	5	5	4	5	5	5
	C	6	5	6	5	6	6	6	7	6
10	B	7	5	7	7	7	7	5	7	5
	C	3	3	7	7	3	4	3	4	4
	A	7	3	5	7	6	5	3	4	3
11	C	3	7	4	6	5	5	5	5	5
	A	2	3	3	4	4	4	4	4	4
	B	5	3	6	6	5	5	5	3	3
12	C	6	4	6	5	5	5	5	5	5
	B	3	6	5	6	6	6	5	5	5
	A	5	3	5	5	4	4	4	4	6

Figure A.18: Data collected from the In-Game Questionnaire inside RobotLife.

			Please rank the 3 visualizations according to how supportive you found them for a positive overall game experience (the game was fun, engaging, immersion, ...) [Third Choice (less supportive)]					
			Please rank the 3 visualizations according to how supportive you found them for a positive overall game experience (the game was fun, engaging, immersion, ...) [Second Choice]					
			Please rank the 3 visualizations according to how supportive you found them for a positive overall game experience (the game was fun, engaging, immersion, ...) [First Choice (more supportive)]					
			Please rank the 3 visualizations according to how well they supported you completing your mission [Third Choice (less supportive)]					
			Please rank the 3 visualizations according to how well they supported you completing your mission [Second Choice]					
			Please rank the 3 visualizations according to how well they supported you completing your mission [First Choice (more supportive)]					
			Please rank the 3 visualizations according to how well they fit into the aesthetics of the game [Third Choice (fits less)]					
			Please rank the 3 visualizations according to how well they fit into the aesthetics of the game [Second Choice]					
			Please rank the 3 visualizations according to how well they fit into the aesthetics of the game [First Choice (fits better)]					
			Please rank the 3 visualization according to how easy they were to read while in motion [Third Choice (more difficult)]					
			Please rank the 3 visualization according to how easy they were to read while in motion [Second Choice]					
			Please rank the 3 visualization according to how easy they were to read while in motion [First Choice (easier)]					
Participant ID:								
1	Visualization A	Visualization C	Visualization B	Visualization A	Visualization C	Visualization B	Visualization A	Visualization C
2	Visualization A	Visualization C	Visualization B	Visualization B	Visualization C	Visualization A	Visualization C	Visualization A
3	Visualization A	Visualization B	Visualization C	Visualization B	Visualization C	Visualization A	Visualization C	Visualization A
4	Visualization B	Visualization A	Visualization C	Visualization B	Visualization A	Visualization C	Visualization A	Visualization C
5	Visualization A	Visualization C	Visualization B	Visualization C	Visualization A	Visualization A	Visualization C	Visualization A
6	Visualization A	Visualization B	Visualization C	Visualization B	Visualization A	Visualization C	Visualization B	Visualization C
7	Visualization A	Visualization B	Visualization C	Visualization B	Visualization C	Visualization A	Visualization A	Visualization C
8	Visualization B	Visualization A	Visualization C	Visualization C	Visualization B	Visualization A	Visualization B	Visualization A
9	Visualization B	Visualization C	Visualization A	Visualization C	Visualization B	Visualization A	Visualization C	Visualization A
10	Visualization C	Visualization A	Visualization B	Visualization B	Visualization C	Visualization A	Visualization B	Visualization C
11	Visualization A	Visualization C	Visualization B	Visualization B	Visualization C	Visualization A	Visualization C	Visualization A
12	Visualization C	Visualization A	Visualization B	Visualization B	Visualization C	Visualization A	Visualization B	Visualization C

Figure A.19: Data collected from the Post-Questionnaire conducted on Google Forms.

List of Figures

1.2	The research context of this thesis lies in the intersection between the Situated Visualizations, Visualization in Motion and Video Games Visualization domains.	3
3.1	Number of occurrences of visualizations in motion per video game genre based on our systematic review.	12
3.2	Results of the systematic review categorization for three dimensions: (a) Data referent, (b) Data Dimensions and (c) Color appearance.	14
3.4	Results of the systematic review categorization for three dimensions: (a) Background consistency, (b) Embedding Locations and (c) Movement autonomy.	16
3.6	Number of occurrences of different visual representation placed in each embedding location emerged from our systematic review.	17
3.7	Number of occurrences of visualizations with different color appearances in different game genres, based on our systematic review.	18
3.8	Number of occurrences of visualizations embedded in the referent in different locations in different game genres, based on our systematic review.	19
4.2	Number of occurrences of visualizations displaying different types of data in different game genres based on our systematic review.	22
4.3	Number of occurrences of visualizations with various visual representation according to different types of data, based on our systematic review.	23
5.3	Screenshot of RobotLife UI. The player's health is displayed on the bottom left part, using a bar chart in red and white cross symbol. The ammunition count is displayed on the bottom right part, using a bar chart in white with a weapon symbol. Primary and secondary objectives of the game are displayed in the left part of the screen, together with the pause button. . .	33
6.1	Visualizations sketched during the brainstorming phase presenting different visual representations, embedding locations, colors and types of encoding. .	37

6.3	The three visualizations embedded in the robots representing the 6 percentages of health level used in the experiment: 18%, 32%, 43%, 58%, 72%, and 83%.	39
6.5	Diagram of the gameplay flow of RobotLife. After the initialization, instruction and tutorial, the game comprised three different blocks, one per each visualization under test. Each block comprised a training, a 5 minutes gameplay of the main level and an in-game questionnaire.	41
7.5	Unity editor with the second level scene opened, showing the level map with the resource spawn point highlighted by the mesh colliders.	52
9.2	Result of the BeauVis [22] likert scale used in the in-game questionnaire to evaluate the aesthetic pleasure of the visualizations studied.	65
A.1	Systematic Review Data - Page 1.	87
A.2	Systematic Review Data - Page 2.	88
A.3	Systematic Review Data - Page 3.	89
A.4	Systematic Review Data - Page 4.	90
A.5	Systematic Review Data - Page 5.	91
A.6	Systematic Review Data - Page 6.	92
A.7	Systematic Review Data - Page 7.	93
A.8	Number of occurrences of visualizations moving autonomously, controlled by the player or depending on both factors depending on the type of data referent based on our systematic review.	94
A.9	Number of occurrences of visualizations embedded in the referent in different locations per each visibility condition (either on focus or not), based on our systematic review.	94
A.10	Number of occurrences of visualizations represented by a specific type of visual representation in different game genres, based on our systematic review.	95
A.14	In game questionnaire proposed inside RobotLife: Likert Scale of 4 items about the visualization's readability.	121
A.15	In game questionnaire proposed inside RobotLife: Likert Scale of 5 item about the visualization's aesthetic. The scale used is BeauVis: A Validated Scale for Measuring the Aesthetic Pleasure of Visual Representations by He at al. [22].	121

A.16 Data about the game performances. A logging system is implemented inside RobotLife and recorded all the salient actions of the player during the experiment. The log files produced have been analyzed to obtain the aggregated data reported in this table.	124
A.17 Data collected from the Pre-Questionnaire conducted on Google Forms. . .	125
A.18 Data collected from the In-Game Questionnaire inside RobotLife.	126
A.19 Data collected from the Post-Questionnaire conducted on Google Forms. .	127

List of Tables

2.1	Different scenarios of visualization in motion identified by Yao et al. [66]. The work of this thesis falls into the scenario highlighted in blue: viewer stationary and visualization moving.	6
3.1	Summary of the visual representations found in the 50 analyzed video games.	13
A.1	List of selected games for the systematic review conducted.	86

Acknowledgements

I would like to thank each and every of my colleagues of the AVIZ team that welcomed me as a part of their beautiful family and supported me during my research journey. Everyone contributed personally to my professional and personal growth, contributing to a valuable and unique experience.

Special thanks to my supervisor Petra Isenberg, who supported and encouraged me day by day. Working with her was a honor and a pleasure. She encouraged my passions, allowing me to have fun while working and contributing to an experience of personal growth and professional enrichment. She encouraged me to growth out of my comfort zone, fully trusting me during the development of the project; to Lijie Yao who welcomed me from day 1 and was always there when I needed help. She was always willing to share valuable knowledge and provide helpful suggestions; to Alexis Pister and Alaul Islam that were always ready to share a laugh and created a lovely office environment; to Mikael Sereno, whose help was fundamental during the implementation or RobotLife and who enriched my knowledge of computer graphic programming; to my advisor France Garzotto, who supported me during my experience abroad, collaborating with my external advisors to complete my thesis.

Lastly, I would like to thank my family and friends that supported me day by day, encouraging my ambitions and contributing to an amazing university experience.

This work was partly supported by the Agence Nationale de la Recherche (ANR), grant number ANR-19-CE33-0012.

