Teaching Space History through Video Games:

Developing an FPS for a Digital Age of Consumers

By

Hirak Choudhury

"This report is submitted in partial fulfilment of the requirements for the M.Sc. degree in Multimedia at the University of Westminster"

Supervisor: Ashif Tejani

14th September, 2016

Multimedia Project-ECMM 799

W1569215

Abstract:

We now live in an age of pervasive computing, where computers control everything in all walks of life.Our live's motto has been changed to be always connected and as we learn and grow with the help of computers, we develop new and radical ways of perceiving the society in which we live.

So why use computer games as a new learning medium? One could argue that they are the most exciting and technically demanding computer applications today with design and development budgets which dwarf any traditional film media production.However, what makes Video games an effective tool for Learning?

The best answer would lie in a quote by Aarseth[1]-

""Playing is integral, not coincidental like the appreciative reader or listener. The creative involvement is a necessary ingredient in the uses of games. The complex nature of simulations is such that a result can't be predicted beforehand; it can vary greatly depending on the player's luck, skill, and creativity.[1]"

Video games are an interactive medium and is perfectly suited for the newer digital generation wherein they can learn and apply knowledge through structured narratives or acquired cues through gameplay.

My aim in this project would be to develop a medium with which gamers can interact and cognitively partake information on historical occurrences.The subject which this project would handle is Space History. These clues which they encounter within the game will further help the to understand the real event and remember relevant information regarding the event opposing to traditional sources of information on that very same subject/matter.

This project will introduce a prototype which is based on a First Person Shooter Mechanic and how effective it is in conveying this information through digital artifacts.

Table of Contents:

Introduction

1.1 Background

Video games based on historical events are not a new concept. It can although be stated that computers are changing the way we learn things. Video games take an active role in this change as they let people participate in new worlds. They let them think, talk and act in new ways[1].As the best form of learning is not by knowing a certain concept but by applying it in a practical situation this way of learning could be the next step in education.

There are many examples of video games which are based on historical events or settings.

Some examples are:

- Sid Meyers Civilization IV: A turn-based strategy game in which the players primary objective is to construct a civilization from limited initial resources.It has been the object of scholarly critique, more than any other historical strategy title.[2]It has a sophisticated combination of civic options. For example, A player can adopt democratic government and state property ownership at the same time.Religion is also a central feature of Civilization IV, it holds the ability to affect an empire's productivity, loyalty and external alliances .Thus it enables students to examine many aspects of history while deconstructing its game mechanics and thematic assumptions.[2]
- Grand Theft Auto: San Andreas: An open world action-adventure game developed by Rockstar North and published by Rockstar games.It is played from a third-person perspective in an open world environment. It is a controversial game depicting cultural similarities to 1990s Los Angeles. The player inhabits the character of Carl Johnson who is placed in a world filled with crime, drugs, and antisocial elements.[3]
- Full Spectrum Warrior is a video game based on a US Army training simulation. The player is forced to think like a professional soldier, the skills and knowledge of the soldier are transferred to the player.

- Assassin's Creed: It is an action-adventure video game series that are set in a fictional history of real world events[4]. The main idea behind the game is reliving an ancestral memory to locate objects of historical significance.

In a marketplace of avid gamers, developers are creating content based on narratives which are often defined by historical occurrences.But how valid are they?..we have got shooters which use active history context like the Call of Duty Series but mostly they do not promote knowledge or educational material. Any Historian who attempts to study "Call of Duty" as a learning medium would soon realize the limitations of the product. At the end of the hour, it is about shooting people rather than actually delving into the historic resource present in the game.
You could ask, Why a shooter to teach historic medium? Well there have been successful studies in the past(most notably the Kurt D. Squire experiment on Civilization series which I would be explaining later), but one must also take into possible consideration the benefits of a first person shooter game to the human mind.
A study by a team of researchers from the University of Rochester claims that Shooters helps in improved decision-making skills without sacrificing accuracy, better pattern recognition-fps shooters can pay attention to more than six things at once without getting confused compared[3] to four that a normal person could keep in mind.Shooters also help in visual attention and cognitive skills.They improve decision-making abilities and also enhance real world perception.

Despite positive findings such as above, prolonged gameplay does have negative drawbacks.Game addiction has not yet been officially recognized as a disorder but certain studies show that five to eleven percent of children worldwide say such games are disrupting their lives[1], suggesting signs of addiction, possible triggers to aggression, introversion, etc.
The Challenge lies in the area to create such a product which would minimize the above-stated factors and also promote the positive aspects of learning through video games.

1.2 AIM:

The aim of this project is to develop and create a multi-platform first person shooter which helps to educate users about the advances in space history over time. This game will intrigue users to discuss the radical developments in space explorations that are happening over the past years. It will also generate interest in the field of astrophysics and promote learning in a fun and interactive way.

1.3 OBJECTIVES:

The MSc project should address the following tasks:

- Investigate existing video games which are based on historical and multicultural events and follow their structure of approach.
- Create its own unique narrative which improves cognition through interaction
- To design and develop a video game based on actual historical events which have occurred in the past.
- To design and develop a unique method of verifying cognitive information presented in the game.
- To minimize the adverse effects of gameplay by proposing a structured narrative which is fixed and independent of player involvement.
- To develop a functional prototype of a single level which demonstrates the above factors and which can be implemented throughout the game.

Literature Review:

Video Games as a trigger to Research:

A study by Nicolas Trépanier titled- "The Assassin's Perspective: Teaching History with Video Games", debates about the inaccuracies of Assassins creed in relation to world history."Assassins creed" are a series of action-adventure video games created by Ubisoft which is set in a fictional history of real world events.The author agrees that video games are efficient in engaging young users to boring age old history. He expects that by learning incorrect representations of history his students would adapt to a sophisticated and critical way of perceiving real historical events. To justify his ideas he presents the following scenarios:

- The author designed a course in which students are required to play a few historically themed video games while they read academic articles related to that game[4]. The Author sorted his collection from a variety of genres because the historical component does not play the same part everywhere. His collection was therefore missed with some adverse titles such as Assassins creed, Total War , Civilization etc.
- Students preferred the latest titles and the ones with a more linear and narrative approach.This, in turn, led them to write research papers based on these historical topics on the latter half of the semester with ranging topics varying from political intrigues in Renaissance Italy to popular ideas of nature in -early-20th-century America, and from conflicting cultural perspectives on the war in the time of Cortés to the motivations behind Viking raids[4].
- By the end of the semester, the students had become keenly aware that the term of comparison when evaluating the reliability of historically themed[4] games, is not real objective history but rather the constantly debated and contradictory outcome of historiographical research.
- The students broke down the concept of historical inaccuracy into a variety of subcategories such as Aesthetics, passive narrative elements, and psychology all while taking into account the unique characteristics of video games as a medium of art expression.

Results:

Including the excitement it created among the students, it also gave them benefits like an alternative approach to engaging with academic publications. The critical perspective that students acquired in this particular seminar might have a more lasting potential than an average history course as the students learned as they engaged in the subject. In this seminar, the video games which the students played before showing up in class provided the basis of their understanding of the historical content.
So this allowed the professor to devote the entire discussion time to the complexity, nuances of the historian's sources methods and interpretation. The level of discussion amongst students rose organically as weeks went by as it was fueled by their interest in the subject. Before long the students were referring to the scope, tone, approach, and arguments presented in the various articles[4].They started invoking considerations about methodological challenges, competing for theoretical approaches, and other ongoing debates. This engagement in histography was significantly larger than ever seen before in undergraduate students.

"Assassin's Creed"does not offer a truthful portrayal of history always but the above study shows how influential it could be to incorporate learning into video games.Most people will be shocked by the idea of a course which teaches through video games in college, and for good reason also. As a medium,video games are subjected to limitations that make them incapable of conveying the complexity of good Histography. Although traditional mediums of learning can never be replaced ,games should complement or motivate students towards the curriculum and therein lies the challenge.

Video Games as a Design Experience:

[Published by Kurt Squire. Titled-"From Content to Context"]

The Author gives a very detailed view on games as a designed experience. He argues that the player's understandings are developed through cycles of performance within the game world which instantiate particular theories of the world.

He talks about these examples in great detail-

- Games as Participation In Ideological Worlds: The Author discusses the outcome of a 2002 Missouri court decision on the legality of restricting access to violent video games which was overturned by the 8th Circuit court with the judge stating that 'video games are as much entitled to free speech as the best of literature.'

  He states that if games cannot represent a point of view, then there's no relative use on the fact that whom the games' are targeted to. To illustrate this problem he gives the example GTA San Andreas where the player is put in the shoes of a black man in the 90s Los Angeles. The game does not require players to run over, harm or kill characters; these are just the choices the player is supposed to make as he progresses. There is a particular ideology at work in the game. The violent streets of San Andreas are rife with gang warfare, and certain actions are rewarded in ways, while others are not even possible[5].
- Knowledge as Performance:

  The author states that today's generation of games contains a whole new set of features, making them intriguing intrinsically motivated sites of learning.He claims that it is important to know what players are learning from games such as Grand Theft Auto, Civilization 3, and Deus Ex 2 as well as their potential in educational media and e-learning. Early studies of games showed how developers used challenge, curiosity, control, and fantasy to engage players[5]. These similar design features have been used to increase learning on pre-post gains in controlled studies using mathematics software(Cordova and Lepper, 1996).Such studies[5] suggest the promise of deriving learning principles from games but are focussed on a relatively general level of abstraction and do not account for innovations from the generations of games such as interactive learning, collaborative problem solving, and game players as producers(Squire 2003).

  As with new games comes new features which make them intriguing sites of learning.We can look at learning both as an interaction in the social and material world and participation in distributed social organizations.James Paul Gee[8](2003, 2004, 2005) argues that video games are an ideal laboratory for studying learning principles because as the games increase in complexity, game designers embed structures to help players learn them. Examining these features may provide insights into the design of other learning environments like educational software, virtual worlds etc.

- Learning by doing: A core characteristic of games is that they are organized around doing[5]. They are uniquely organized for a functional epistemology, where one learns through doing, through performance (Squire, in the press a). Cognition in digital worlds is thoroughly mediated by players' capacities for action: The player's actions are his or her interface with the world [9].

  Perception of the game world is the other half of the perception/action system.Games' graphics are more than pretty pictures, they are signs that the player must learn to read.As players interact with the world to ascertain possibilities for action, they develop a professional vision for the affordances of the world[5](Gee, 2005; Goodwin 1994; Jenkins and Squire, 2004). Through recursive cycles of perceiving and acting, thinking and doing within the game system, a player begins to adopt a particular perceptivity of the character within the game world.He/She becomes a hybrid version of himself/herself playing as the Character like Carl Johnson in San Andreas or the leader of a civilization(Gee,2005). These leads to the player setting up projective identities of themselves in the game world. So the decisions they make in the game are a synthesis between the character and affordances-capacities for the action of the avatar. As a result, the player learns not just the facts or procedures but how to "be " in the world as the game character, also developing appreciative systems of the avatar in the process.

- He classifies games on exogenous and Endogenous based on their contrasting game types.He categorizes each genre and the effect it has on the cognitive ability of a social-gamer and a single-gamer on the basis of Learning, Knowledge, Instruction, Social Model, Pre-knowledge etc.
- He also explains how entertainment games are used in the field of learning .Games such as Sim City, Civilization, and Tycoon simulation games have been hypothesized by researchers that they could be effective learning tools. Students used these game experiences to think about why civilizations grow, flourish, and fade, and how wars, revolutions, and civilizations evolve as the products of interweaving geographical, social, economic, and political forces[5]. As students interacted with the game and discussed it in class, they began to understand its ideological bias and at times, took it up as a framework for explaining world history

He concludes by stating that although digital games have largely been ignored by educational researchers[5], they are a powerful new medium with potential implications for schooling. In video games, [5]knowing is at its essence a kind of performance, as learners learn by doing, but within powerful constraints instantiated through software and social systems. The focus is on experience that enables students to develop situated understandings, to learn through failure, and to develop identities as expert problem solvers (Gee, 2003; Squire, in press a).

Video Games as Writing History Through Algorithms

[Published by Shawn Graham titled- "Writing History through Video Games"]

The author writes about making video games or more accurately learning to depict them as in history-through-algorithms.

The research was conducted on undergraduate students(history majors) of Carleton University who were asked to explicitly write their own algorithms to generate particular kinds of emergent engagement through historical materials.This approach is similar to a reader who makes the story in the process of "reading". The author quotes

- William Urrichio(2005) who was not concerned with the graphical representations of the past but rather with the ways the gameplay allowed for different understandings of history to be represented[1].
- Ian Bogost[2007] who coined the phrase "procedural rhetoric" to express the above thought defining the idea that the processes of computation embody a kind of rhetoric and representation of how the world is also a kind of cosmology and how one can learn a lot about game designers just by reading their code.

The Assignment :

The students were assigned one major project to complete over the duration of their course: To design an ideal game on basis of their judgment on how it would look/feel/behave. The assignment prompt was:

"In small groups (assigned by the instructor), you will produce a 40-50 page game design document for an ideal history game (or meta game; a game about games) that distils what you have learned about telling history through interactive media. This document will also demonstrate in passing what you have learned as a result of this course.You will need to

reference the appropriate games, history learning, games, and history,design, psychology,cognitive science or other literature to explain and show how your game/simulation would achieve its desired ends.For the purposes of this course, you do not need to produce the actual game. Although, you may wish to create a playable mock-up or 'beta' of what the game might look like. It should demonstrate key concepts or gameplay mechanics, and be about 10 minutes worth of play . If you create a mockup along those lines, your written document can be correspondingly shorter."[11]

Numerous checkpoints were established to keep a project of such scale in check.The first checkpoint was the "pitch" where the students were constrained to describe their ideas in a single short paragraph. The next checkpoint was to identify the "problem-space" and the principle game mechanic for addressing this space.The problem-space are challenges that the player is expected to overcome during gameplay.The key here was to think of history filled with problem spaces to force the student to think of how actors in the past were "confined by resources and rules of interaction with others[11]."

The final checkpoint described in detail the player's experience at each stage of the game, that is again what they feel/learn/do in the game.Writing algorithmically is about writing spare,lean code and use minimal expressions to get the job done so these students did not write code per se , they wrote academic code in a way which mimicked computer code.

The Outcomes:

At the end of the course, there were six group projects submitted, out of which only one project titled "The Medics War" fulfilled the objectives set by the author[11].The creator's quote-

" Our [world war I] game looks to broaden the emotional range of video games and the players.[11]W e strive to illuminate the tragedy of war by creating an empathy with a group that has not been explored yet – the field medic.Many games that show history are focused colonizing, on conquering, about playing at war [11] Our game doesn't rely on domination but rather attempting to show the true nature of war; no matter which side you're playing on, there will be casualties, soldiers who are following orders, that need aid. [11]"

Conclusion:

This study provides ample details on how by encouraging students to use technology by learning how it is produced, they are able to look back at how the history is constructed by

reflecting themselves within the game. This process enables them to engage with the storyline, the events of the past reconstructed in the game, and when they are able to immerse themselves into the game, they absorb the facts and repercussions of the past without concerning much about the minute details[11].Thus engaging them on a personal level with the past.leaning to play while they are playing to learn.

This study proves that short challenges like the above example could change the way we learn things.This method could enable us to create prototypes which not only helps us understand historic materials effectively but also on successful creation could spread knowledge and create new narratives through video games.

Action Shooters as the new Age Nootropics

[Study by Lydia Denworth,2013 titled:Brain-Changing Games]

Violent First-person-shooters or action-shooters, in general, have received negative criticism due to the fact of its mindless violence genre.Scientists are trying to figure out how and why these games affect players to create products that emphasize benefits but have fewer drawbacks[12].

Neuroscientist Daphne Bavelier of the University of Rochester and the University of Geneva conducted a study with which she would deduce how the brain responds to Action Shooters by re-tuning connectivity across and within different brain areas.

The Experiment and it's outcomes:

Bavelier assigned an undergraduate, C.Shawn Green, to program his own version of a standard test of visual attention in which individuals would first identify a central white square and then indicate on a touch screen the location of a shape that briefly flashes some distance away. The task is known as -"useful field of view", measures spatial attention- that is, the ability to keep track of multiple locations and shift attention across space[12]. This ability is used in day to day activities of multitracking though the human eye. For example, while driving we employ this skill as in the rate at which our focus could shift to a sudden movement in the right. Green tested it on himself and surprisingly he did twice as well as the normal rate. He faulted his programming but brought in some friends to test it further[12].They also scored as high as Green did.

This prompted Bavelier to take the test.She fell within the normal range which compared to the previous average was poor.The reason was due to the fact that Green and his friends were avid gamers of action-packed, fps genres.

Bavelier reassigned Green to a new study that compared various aspects of visual attention in eight action gamers and eight nongamers[12].

- In one task subjects were asked to report the number of squares flashing on a screen at a single time.The more items a person could register immediately, the greater his/her attentional capacity.Gamers averaged 4.9 items versus 3.3 for non-gamers.[1]
- In a test of attention to locations in space,gamers were roughly twice as accurate as non-gamers at indicating where targets appeared
- Gamers also significantly outperformed non-gamers when they had to identify, and thus pay attention to ,whether certain letters appeared in a string of letters flashed in rapid succession[].

With positive results such as the ones stated above, Bavelier decided to approach action video games more methodically.She tried to access the result of games on visual acuity.To perform this experiment ten gamers and non-gamers were asked to determine whether the letter 'T' was right side up or upside down like other similar 'T'-shapes crowded in.They measured how close together the letters could be before interfering with performance. This ability is the skill which could distinguish detail in cluttered environments.The former is critical for reading.As expected, gamers could tolerate more crowding and still pick out the 'T' , suggesting their detail detection was better[12]. In other findings[12] Bavelier and her team demonstrated that gamers also have better contrast sensitivity, which could be useful is distinguishing layers in fog and is also a necessary skill for radiologists.

Playing shooters could also improve certain visual disorders like amblyopia,or "the lazy eye",blurred or poor vision in one eye disrupts neuronal circuits in the visual cortex during development, leaving one eye underdeveloped.In children, doctors patch the dominant eye to strengthen the weaker eye[1w]. Yet this treatment does not work for adults. In 2011 research optometrists Roger Li and Dennis Levi[12] and their colleagues at the University of California, Berkeley, published a study in which ten adults with amblyopia played "Medal of Honor"- a first-person shooter, for 40 hours with one eye patched.Three other patients played a non-action video game, and seven had their eyes patched before play

began.Tested before and after training, patients who played one-eyed saw their acuity improve more than 30 percent[12], which is a five-fold greater recovery that would be expected when patching on children. In addition, the adult's spatial attention skills got a 40 percent boost,their depth perception increased by 50 percent.

With such positive findings, researchers considered what other brain processes they could tweak.Some notable mentions are-

- Cognitive psychologist Ian Spence's (University of Toronto) research on why males tend to perform better in tasks such as Spatial Reasoning, Field of View and mental Rotation[1] : a group of 48 university students was divided into two teams each of 6 male and 14 female students. None of them were gamers.Each team was divided into sub-pairs of the same sex. One member of each pair trained on 10 hours on "Medal of Honor: Allied Assault" while the other played Balance-a three-dimensional puzzle game involving steering a ball through an obstacle-laden maze. After this experiment, the shooter players improved by 10-15 percent on both the field of view tasks and a mental rotation challenge, whereas the puzzle-game players saw no change. In either instance, the females improved the most, virtually erasing the gender disparity in field of view, and significantly reducing it in mental rotation[12]. Also notably playing these games can sharpen both types of spatial acuity and therefore perhaps even scientific aptitude.
- A study by Cognitive Psychologist Christopher Sanchez(Oregon State University) connected game-induced improvements in spatial reasoning with the ability to learn certain types of scientific material[12].

  1.Participants played 25 minutes of either "Halo", an interstellar first person shooter, or "World Whomp",a timed spelling game.

  2.Next, they read a brief non-scientific text as a diversionary task followed by an explanation of plate tectonics.

  3.Finally, they were asked to write an essay on the causes of the eruption of Mount St. Helens[12].

  Those who played Halo scored better on the essay than those who played World Whomp as measured by their knowledge of five facts about plate tectonics.Spatial reasoning also improved after playing Halo[12].This happened

because the players of Halo had a better mental image, a three-dimensional model that was dynamically changing inside their heads.

- Video games also train hand-eye coordination, although the primary improvement in this domain appears to be cognitive[12]. A study by neuroscientist Lauren E.Sergio (York university, 2010) scanned the brains of gamers and non-gamers (13 of each) using functional MRI while they performed increasingly difficult hand-eye tasks while looking at a screen. The easiest tasks were those in which a person could watch a target, such as pressing a tab on the screen, followed by those that required a user to look away from their hands similar to using a mouse to operate a computer[12]. In the most complex tasks, participants were asked to move the joystick in the opposite direction of the stimulus-inverting the axis of the joystick-meaning they had to inhibit the natural tendency to follow what the eye sees. The harder the task, the more it recruited the prefrontal cortex.

    The gamers were observed to be using fewer motor-control parts than their peers.The other areas were more active, mainly in the front part of the brain.The difference was greatest on the most difficult skills, such as those that involved acting in a manner disagreeing with that of a cue.

These results, according to Sergio suggest that gamers use their prefrontal cortex to perform visuomotor skills more than non-gamers do. This pattern, often seen in musicians, is considered a sign of expertise and may lead to better performance during extremely complex motor feats such as playing piano or surgery[12].

Co-incidentally, a striking application of this skill surfaced in a 2007(Douglas A. Gentile, Iowa State University) study of 33 laparoscopic surgeons[12] where the top surgeons were avid gamers in the past.

Conclusions:

Despite positive findings, heavy use of video games have some serious drawbacks. A study by Gentile found that[1]:

- 5 to 11 percent of children consider games are disrupting their lives making them, addicts. In the United States, the figure was 8.5 percent.But in contrast, 4 to 6 percent of casino gamblers are considered addicted. Thus, even if violent video games can be beneficial, people need to be alert to the dangers of too high a dose.

- Video games lead to a short-lived increase in aggression although involved parenting and good social skills can minimize the problem.

- Games change the way kids see the world[12].

Researchers would be able to tease out the beneficial ingredients of these games to create non-violent versions that train the brain just as effectively.So far the above factors are only applicable from a first person point of view, managing multiple streams of information and goals and making rapid decisions[12].

What makes games fun and absorbing are rich graphics and complex plots.

"The very mechanics that seem to make commercially successful games super-fun are also the ones that are seeming to have the positive effects in terms of brain plasticity," says Alan Gershenfeld[12], president of E-Line Media, a company he co-founded to create games for learning, health, and social impact.

Hence, the success of building a new state of brain changing games will require partnerships between neuroscientists and expert game designers.The challenge would also be to keep gameplay limited between sets and create independent and manageable chunks of information which could possibly be devoid of the above limitations.

Design and Implementation

The following list represents the design issues considered in developing the first prototype.

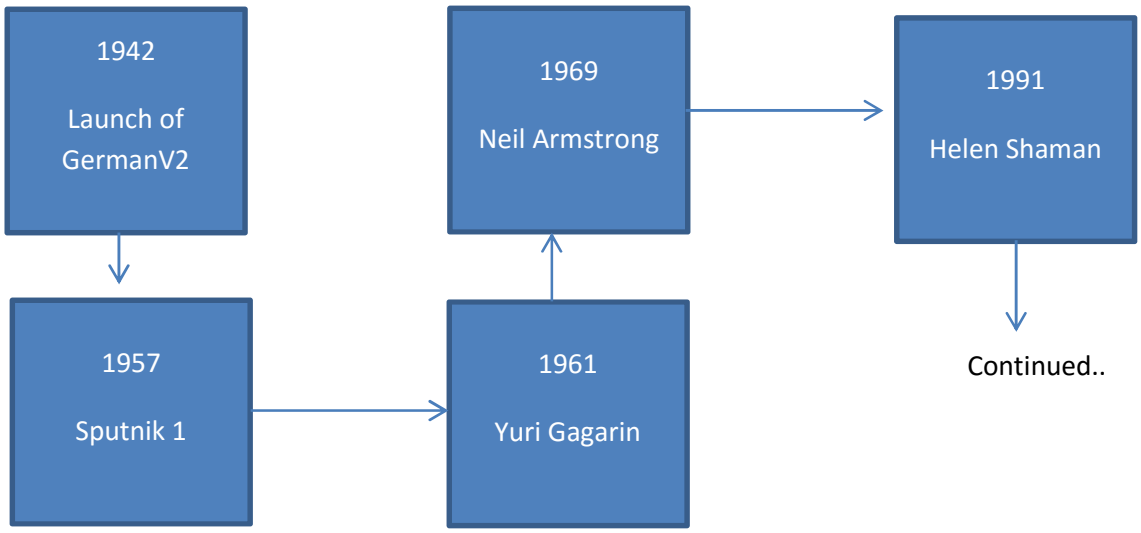
Issues:

- Consistency: Color scheme, textures, and buttons in the game are kept consistent which will ease in learnability
- Simplicity: Keep the User Interface clutter free and display only the required information.
- Adaptability: Support for different preferences for different users.Professional gamers may get a challenge to complete a task whereas new gamers will have an easier option.
- Provide Feedback: Each interface within the game will be clearly defined and users will get relative feedback on each action performed/decision taken.
- Support Navigation and flexibility: The game's User Interface will be well tied together with no loose ends and players will have the freedom to navigate/quit from any part of the game.
- Support Predictability: Each interface object would behave as expected.

Voyage Game Design

Voyage is a first-person shooter which enables players to learn space history by interacting with visual clues and implementing them within the game environment.It follows a linear narrative structure where the player is presented with a crisis and is urged to solve the mystery behind this particular crisis. The levels in the game represent the space timeline with each event in history corresponding to an in-game level.

Once in the game, the player takes the role of a hacker whose objective is to unlock information hidden inside a foreign authoritarian computer.He progresses as he collects memory fragments throughout the course of the game.The timeline followed by Voyage is roughly represented by:



Here each of these events corresponds to a game level which has its unique level design and properties. As a prototype, only the first event is developed but this timeline would continue up to the current event in space history.

The purpose of this game is to integrate learning with gameplay so that the player takes an active interest after he completes each event to pursue further information on the particular topic.

How it works(Voyage Specifications)

The main rules for the game are as follows:

- The player is equipped with a weapon which is a thematic representation of a delete button.It allows the user to interact with UI present inside the level and also eliminate firewall entities.

- The level is actively patrolled by firewall entities, let's call them enemyAI, whose main purpose is to delay the progress of the hacker who is trying to reach the core memory.

- The level is scattered with blocks of code.The player is supposed to collect them. These code blocks are the representation of the program the hacker is writing to unlock the memory.As the code blocks increases, the program gets more robust and the firewall weakens.

- The level is also scattered with fragments pertaining to the original memory. These are UI objects set in the level at various locations hidden from sight and guarded by enemyAI. The player is supposed to collect these fragments and read it through.

-The In-game UI of Voyage has several components:

1.The main timer: It represents the time the user has to finish the level or complete the hack process.

2.Player Health and Ammo levels: These indicators show how much ammo or health the player has, these are essential indicators as they define how much damage the player can deal and how much he can take.Once the player is killed his hacking progress is restarted from the initial checkpoint of the level. This wastes time and can be critical during certain situations.

3.Codeblock count and Fragment Count: These UI elements give a representation of the number of code blocks and fragments the player has collected or interacted with.

4.The objective panel: This panel defines the objectives the player is supposed to follow to clear a particular level.

- The prototype level is cleared when all the fragments present inside the level are downloaded and all the code blocks set in the objective panel are collected. Upon completion of the objectives, the player is required to upload the code blocks and the fragment to gain access to the memory.

-The player then enters another level which acts as an obstacle to his goal. It tests the player's ability to retain information which was presented earlier in the previous level. If the player succeeds in this cognition test then as a reward, he unlocks the memory.

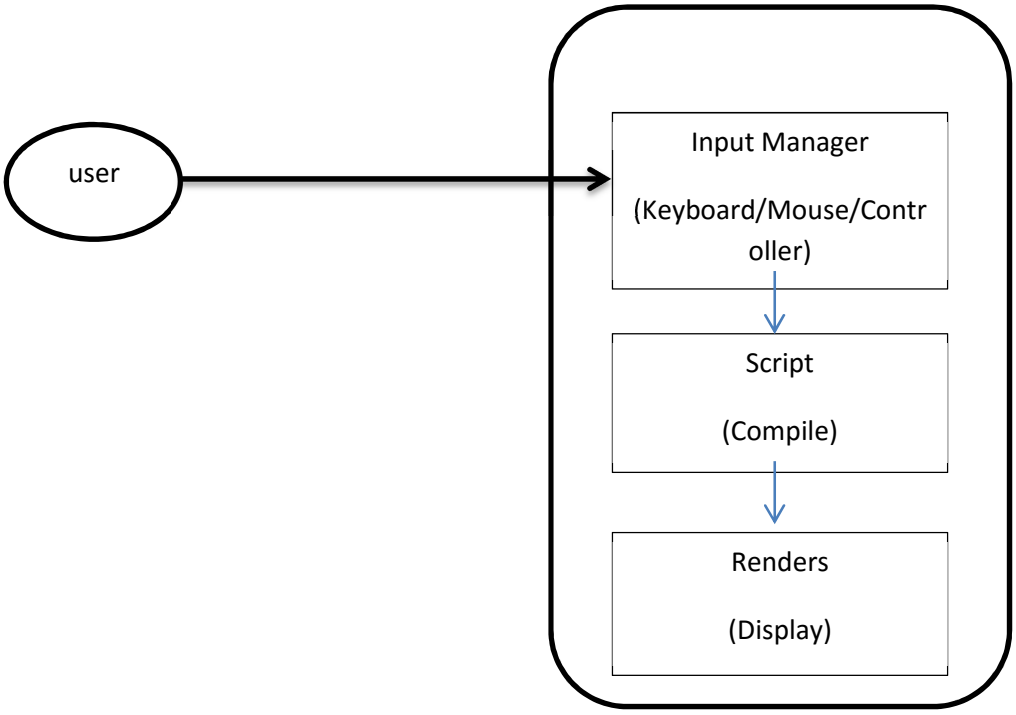
-The memory could range from a video file which shows relevant information to a custom created animation which explains the event in detail.

General Description

This section includes the perspective of "Voyage." and the system environment it requires.It also specifies User story of it.

G.1 System Environment

The user can interact with the prototype by giving input to the system.The System will forward these inputs to scripts, correspondingly if any change of value occurs it will implement those changes via the display as output to the user.



G.2 User story of Voyage:

After running the Game, the UX view of the game appears on the screen.The term UX(user-experience) is used to explain all aspects of a user's experience with the system.The user will have two options to choose from, "Play" and "Exit.". The user can directly click on "Play" and start playing the game, once his/her initial playthrough is complete, he/she will have access to a third option in the main menu titled-"Level Select".This option will allow the user to switch levels or replay a specific level once it is unlocked.

[Fig]

While in the game, the user has access to a "Pause" menu by pressing the "Escape" key on the keyboard. This menu allows the user to pause gameplay at will.The user can either quit the application or restart the current level he/she is at the moment.

[Fig]

If the main Timer(in-game UI) runs out or if the user fails in the bonus level, the user loses the level. He/She can then enters the game over menu.It has two options which enable the user to either retry the level or exit the application.

The narrative starts with a middle-aged man who is fed up with the world that he is living in.The society around him is aloof and have lost the ability to communicate with one another.Humans behave like machines filling their role in the cogs of a higher authority.No one knows what or who is controlling them. The Protagonist decides that to know what has caused this change he must decipher the source of what caused this dismay in the first place.He infiltrates a secret base which contains all the hidden information about this change. He activates the master computer controlling the base and begins hacking the controlling server. As he begins hacking, he is drawn into a digital realm where he must fight to gain information and thus begins the game.

The initial idea was to incorporate a narrative in the form of a graphic novel. This novel would play at the loading screen between consequent levels to give the player a perspective of his progress. Due to time constraints, this is partially implemented in the prototype.

Voyage Use-Case Scenarios

V.1 Use-case diagrams with Case Description

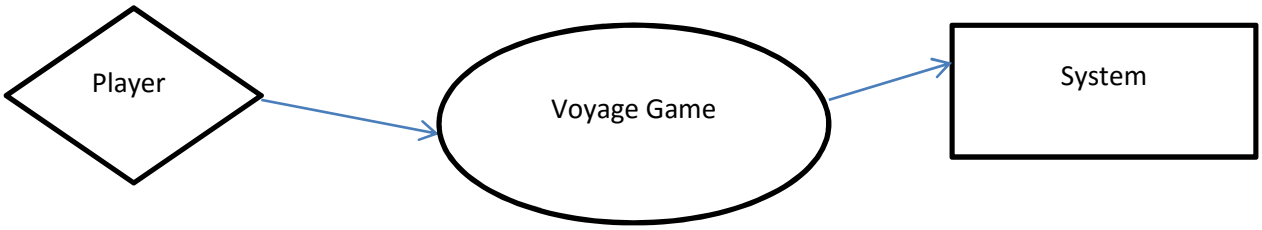


Fig 1: Level 0 for GameUX

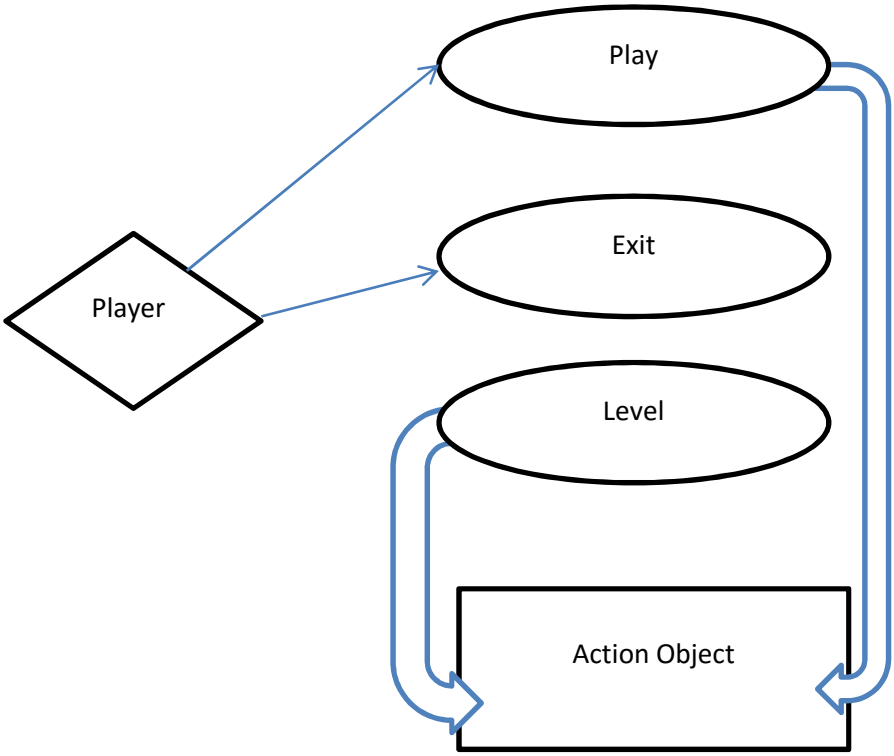


Fig2: Level 1(Main Menu) for GameUX

i.

Use Case: Play

Primary Actors: Anyone playing the game.

The goal in Context: To start a new game.

Precondition:

1.The system supports the game configuration.

2.The file has been triggered to run, and the game splash screen has appeared.

Triggers: The player needs to start a new game.

Scenario:

1.The introductory splash screen or main menu of the game.

2.Click on the play button.

3. The Game is loaded on the system.

Exception: Game Crashed.

Priority: High Priority.Must be implemented.

ii.

Use Case: Level

Primary Actors: Anyone playing the game.

The goal in context: To load the game from a specific level.

Precondition:

1.Required level is unlocked.

2.The game supports loading levels.

Triggers: Need to load a level.

Scenario:

1.Go to the main menu or the splash screen.

2.Click the level option.

3.Select a level.

4.The level is loaded for play.

Exception: Level cannot be loaded.

Priority: Essential, must be implemented.

iii

Use case: Exit Game

Primary Actors: Anyone playing the game.

The goal in Context: To exit the prototype.

Precondition: The prototype application is loaded, or a level is being played, or the player is dead.

Triggers: Player needs to exit from the game level or the application itself.

Scenario:

1. Press Exit either in application or pause menu or game over menu.
2. The Game is exited to the system.

Exception:

1.Something went wrong.Cannot exit now.

Priority: Essential, must be implemented.

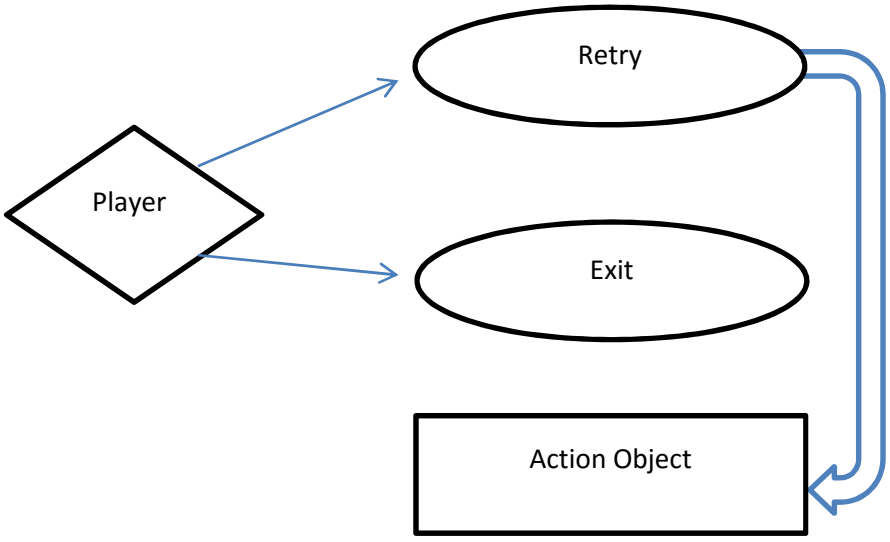


Fig3: Level3 (In-Game Pause Menu) && Firewall Breach Failed/Game Over Menu

lv

Use Case: Retry(In Pause Menu)

Primary Actors: Anyone playing the game.

The goal in Context: To reload the current scene.

Precondition:

1.Player has loaded a specific level and is playing that level

2.Player is in the pause menu or has paused the game.

Triggers: Player needs to restart the level.

Scenario:

1.Click on the retry button

2.The specific level is reloaded with initial values.

Exception:

1.Load state failed,cannot load level

Priority: Optional

lv

Use Case: Retry(In Game Over Menu)

Primary Actors: Anyone playing the game.

The goal in Context: To reload the current scene or the previous scene(in case of cognition level).

Precondition: 1.Player is dead, the timer has run out.

2.Player has entered the cognition level and is dead.

Triggers: Player needs to restart the level.

Scenario:

1.Click on the retry button

2.The specific level is reloaded with initial values.

Exception:.Load state failed,cannot load level

Priority: Essential, must be implemented.

Data Model

Any software prototype requires the need to create,extend or interface with a database.Although Voyage has many data objects, it does not have any data storage.All the objects and their related data are handled by the Unity Game Engine.For this reason, data model is redundant for this game project.

Behavioral Model of the Voyage Game

The behavioral of "Voyage" indicates how it responds to external events or stimuli. It could be represented in two ways; one is creating a state diagram and the other is a sequence diagram. To keep things simple, the state diagram presented here in the report will portray the whole system rather than for individual classes. The modules of the use case scenarios are used to create the state diagrams. The Sequence diagram, on the other hand, is created for all the use cases possible.

State Diagrams:

From Level select to level complete and in Game Menus.(P.T.O).The State: X can be any level from the introductory level to other consecutive ones,

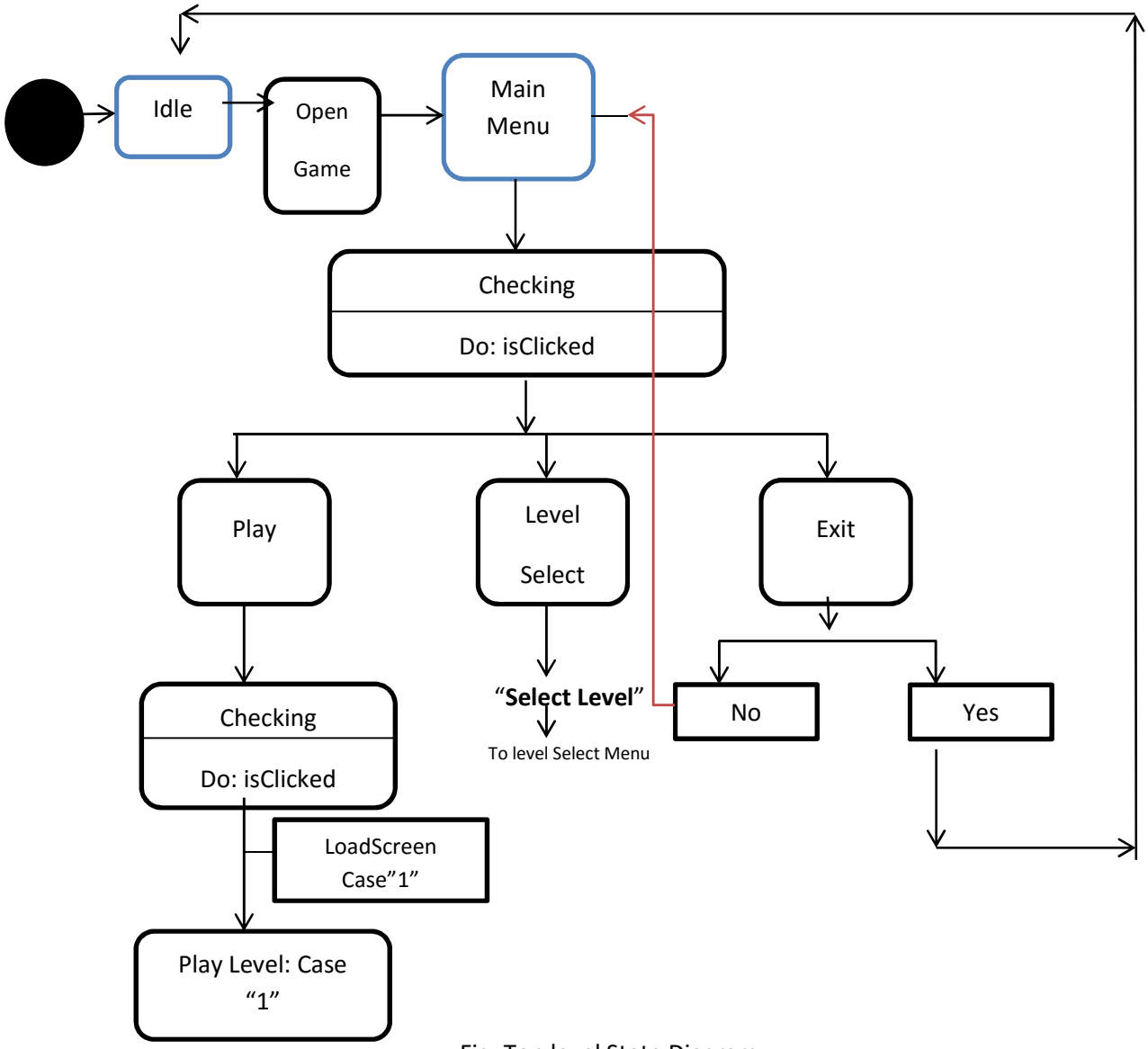


Fig: Top level State Diagram

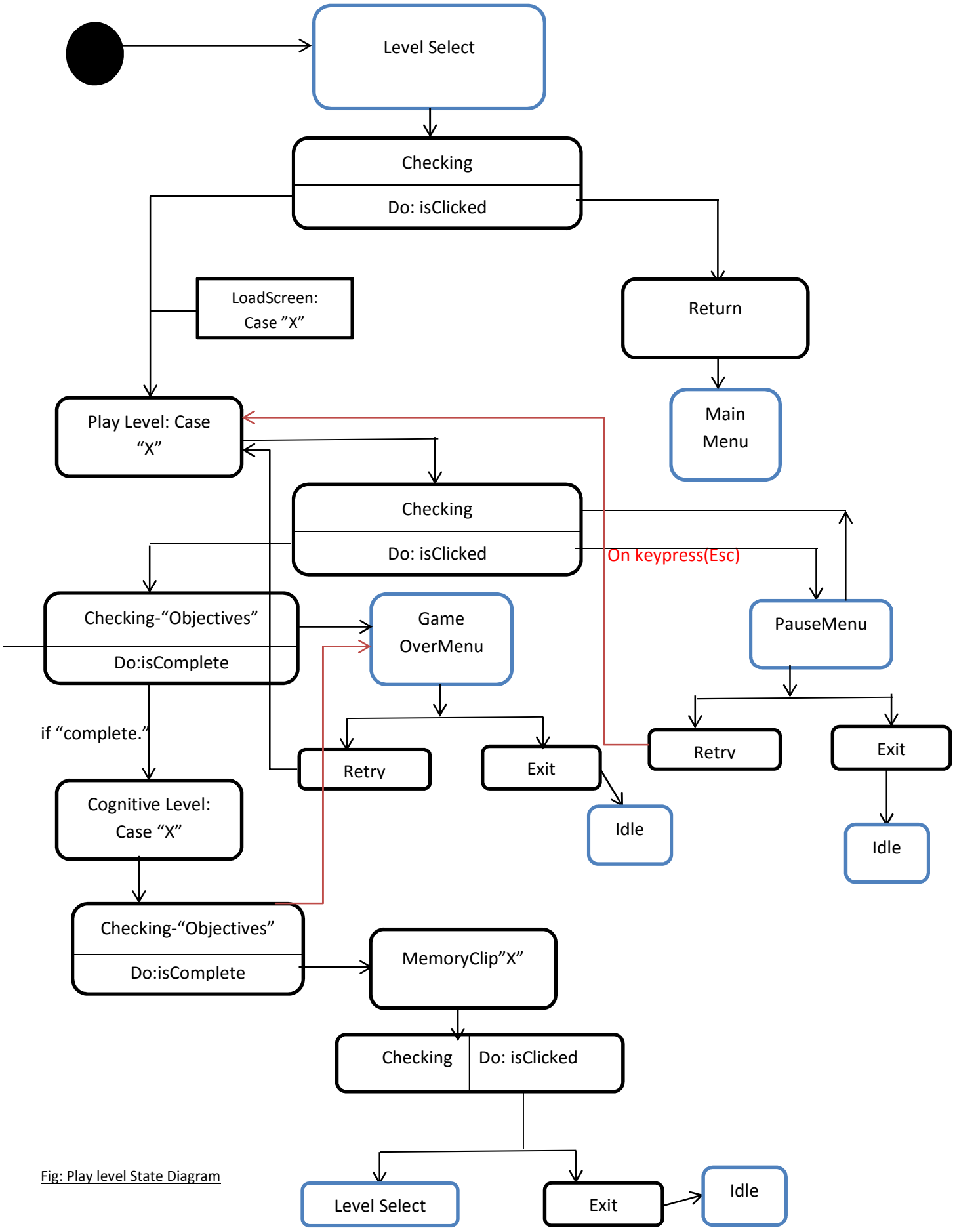


Fig: Play level State Diagram

Sequence Diagrams:

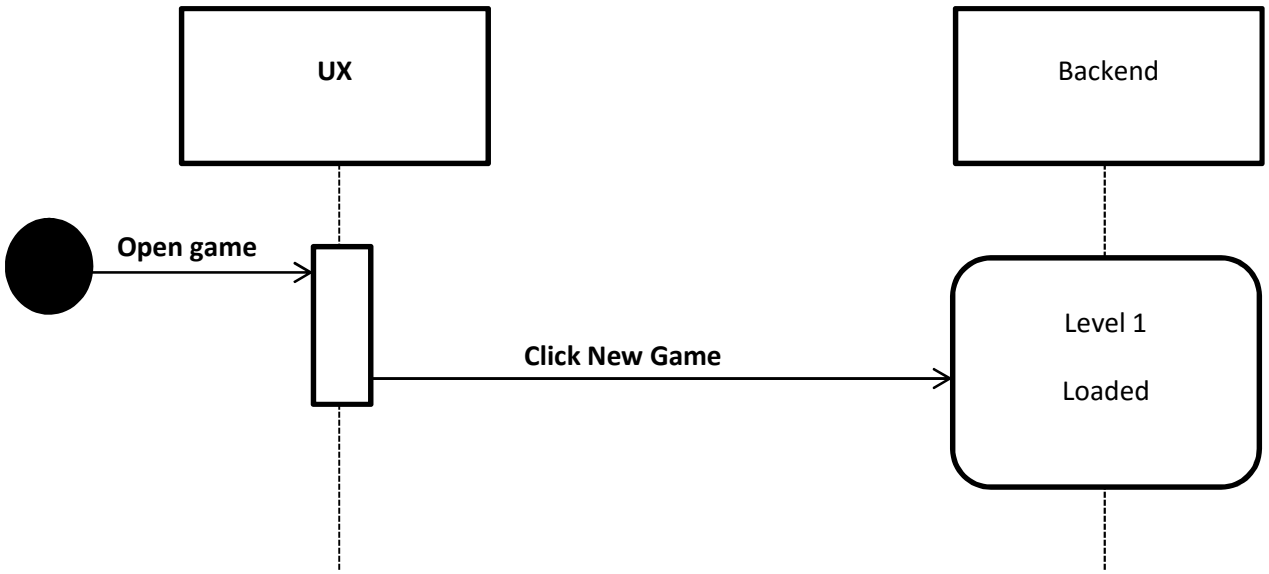


Fig: Sequence diagram(New Game)

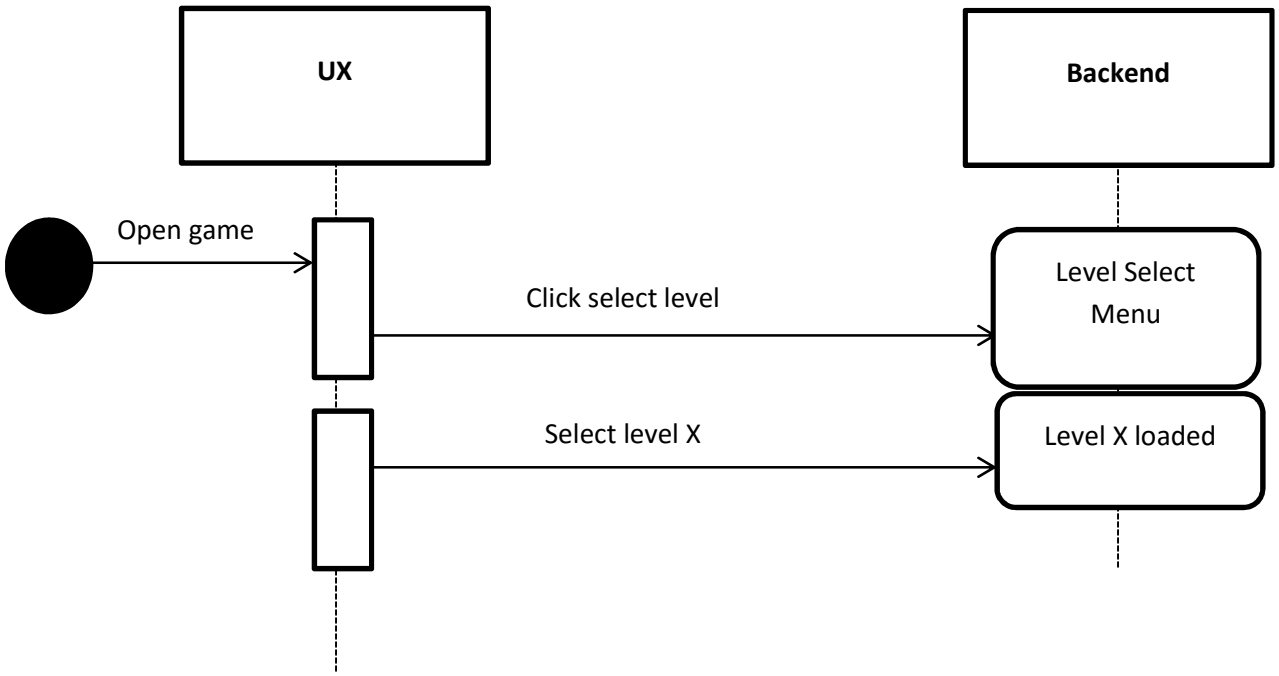


Fig: Sequence Diagram(Select Level)

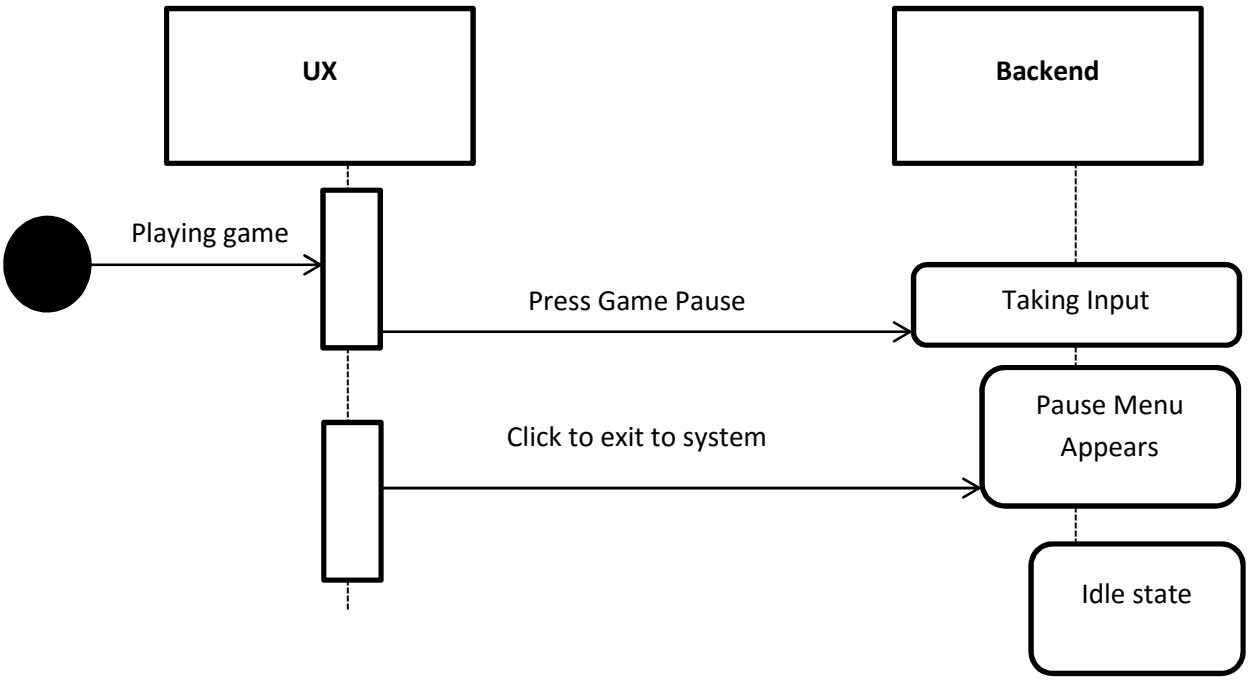


Fig:Sequence diagram(Exit Game)

Development and Implementation

(Sub).1 Methodology

The scope of the proposed game required much research into areas like object-oriented programming and in languages like C#. In limited time, it was not fully ideal to implement a RAD or an agile methodology, as I had to start everything from the ground-up. Hence I chose the traditional waterfall model of development where I allotted milestones for each checkpoint in Game development.

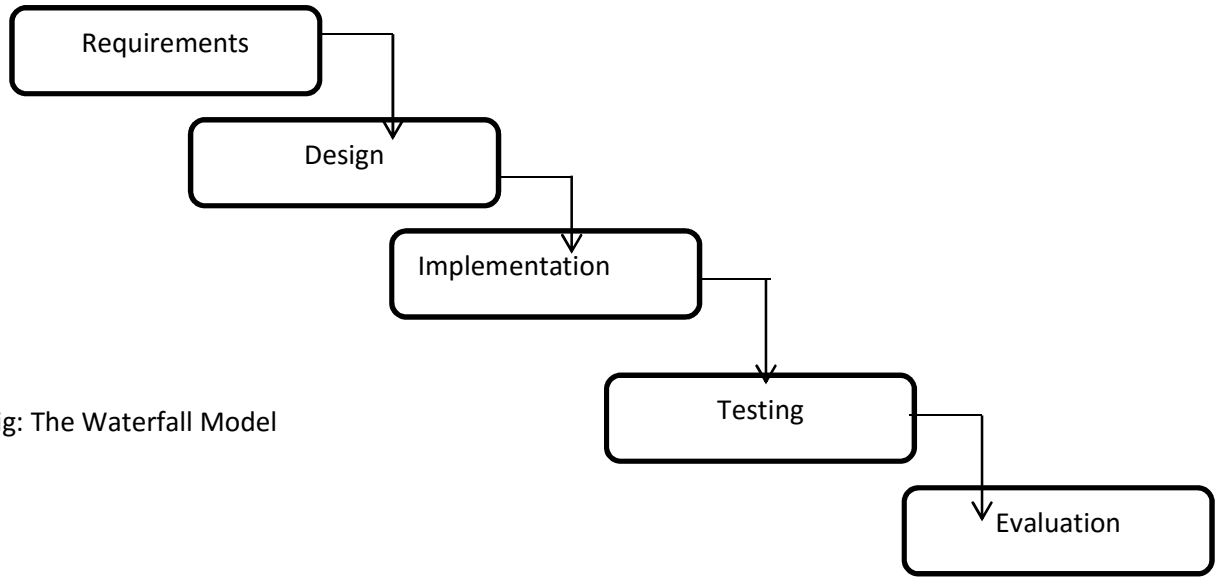


Fig: The Waterfall Model

(Sub).2 Milestones

The development process of the game was divided into milestones to be able to finish the prototype in time.These milestones were built to keep a regulation in what has been developed and what needs to be developed on the previous milestones.The development process of Voyage was constructed like this:

Milestone 1: Establishing the core mechanics of the game that would remain constant throughout.

Milestone 2: Designing assets required in the game and replacing them in the playable level.

Milestone 3: Integrating scripts and defining AI coroutines in the game.

Milestone 4: Creating in-game User Interface and User Experience elements.

Milestone 5: Testing the game under different conditions and creating a level which evaluates information gained from playthrough.

Milestone 6:The final milestone was to tie everything together to form a fully functional prototype which would work in different platforms.

(Sub).3 Tools

Coming from an Animation background, I had used quite some Animation tools in the past few years.Game development was something I wanted to learn and integrate my previous skills of Animation with. Hence, my research was particularly confined on learning Object Oriented programming in the Unity Environment.I chose Unity, as C# was a moderately easy language to learn and the web help in Unity is easier to locate, and very well defined compared to other game engines like CryEngine or Unreal Engine. Also, the hardware requirements for running the other engines was high compared to Unity.In the following, I will demonstrate a quick review of the tools above.

1.Unity

Unity is a cross-platform game engine developed by Unity Technologies and[13] used to video games for PC, consoles, mobile devices and websites.With the new support for HTML5 integration it can run on any browsers eg: IExplorer, Firefox, Opera, Chrome, Safari etc. It has its own web player built but there are only a few browsers which support it.Due to the new HTML5 built it has been losing popularity as the web player does not offer direct streaming of data as HTML5 does. Unity is free for standalone application development.It targets the following APIs: Direct 3d on windows and Xbox consoles.OpenGL on Mac, Linux, and Windows, OpenGL ES on Android and iOS and proprietary APIs on video game consoles.

The Voyage Prototype is currently developed only for the Windows and the Unity WebGL version. Developing in Unity was a challenge because it was a different environment to work with but by the end of the development phase I was well familiarized with the environment, it's scripting libraries and assets, and I could manage to finish the functional prototype in time.

2.Autodesk Maya

Maya is a 3d computer graphics software that runs on Windows, OSX, and Linux developed by Autodesk, Inc.It is used to create interactive 3D applications, which includes video games, animated films, TV series, or visual effects[14].

I used Maya mostly to create reference models for the Voyage.The "DeleteGun." or the weapon used in the game was modeled and texture-mapped from scratch and was the most time-intensive.The Enemy AI was also modeled and rigged in Maya and exported to .fbx format for importing in Unity.

Some of the secondary assets used in the prototype were imported from the asset store directly in the project. Example: Game Concept Pack. These were mostly primary meshes such as walls, floors, etc.

3.Texturing Tools(Adobe Photoshop and MS Fresh Paint)

Adobe Photoshop is a raster graphics editor developed and published by Adobe Systems for Mac OS and Windows[1].MS Fresh Paint was initially launched with Windows 8 in 2012.It simulates the behavior of physical oil and water paint on a digital medium.

Both the narratives of Voyage and the textures in the game was designed on MS Fresh Paint to give a thematic similarity to the application.These textures were then mapped accordingly to the UV Snapshots generated by Maya.

4.Adobe After Effects

Adobe After Effects is a digital visual effects, motion graphics and compositing application developed by Adobe Systems and used in the post-production process of filmmaking[15]. In this context, AE was used to create loading screens between level and to integrate the graphic panels and create a structure to the narrative.It has been partially implemented in the prototype as well.

5.UFPS plugin for Unity

UFPS is a professional First-Person-Shooter base platform plugin for Unity. It is known for smooth controls and fluid realtime-generated camera and weapon motion.UFPS helped in speeding up the development process as it already had pre-generated scripts for camera and weapon movement which was both professional and easy to implement. However, implementing UFPS came with some limitations such as conflicting scripts, and almost minimal support for publishing on mobile platforms.

Product Design Terms

This chapter covers the product design phases and the system features of Voyage.

For every enterprise product two key terns of design are very important.They are:

1.User Experience(UX)

2.Backend Programming

(sub).1 User Experience(UX)

User Experience design(UXD or UED) is any aspect of an user's experience with the given system, including the interface, graphics, industrial design, physical interaction, and the manual in most cases.

User Experience Design fully encompasses traditional Human-Computer Interaction (HCI) design, and extends it by addressing all aspects of a product or service as perceived by the users[16].

(sub).2 Backend Programming

The "back end" portion of a program is that piece of code that supports the front end(UX).It is mostly responsible for database access, business logic etc.

For efficient implementation and to increase user acceptance of the product both are very important in the software industry.

(sub).3 System Features of Voyage

1. Title Menu
2. Level Menu
3. Pause Menu
4. Game Over Screen
5. Main Timer
6. Codeblock Count
7. Memory Fragment Count
8. Codeblock Pickup
9. Health Pickup
10. Ammo Pickup
11. Floating Fragment UI
12. Enemy AI

13. Upload Codeblocks Interface
14. Upload Fragments Interface
15. Automatic Doors, Platforms, etc
16. Exit Points
17. In-game Cognition Level
18. Logic tiles
19. Main Memory Pickup
20. Main Memory User Interface

(sub).3.1 Main Title/Splash screen

(sub).3.1.1 Description and Priority

The title screen is what the player sees every time upon playing the game.This interface has 3 options- Play: To start a new game, Level: To select Levels, Exit: To quit the interface.The title screen is the "hub" for all activities in the prototype.

(sub).3.1.2 Stimulus/Response Sequences

Step 1: The player starts the game from either the game website or with their PC.

Step 2: Unity loads the resources or components(Only in the case of Web built) and then the title screen pops up with three options available for interaction namely: "Play", "Level", "Exit".

Step 3: The player presses one of the buttons which will trigger a function on response.

(sub).3.1.3 Functional Requirements

REQ 1: The title screen must load and appear every time the game is launched.

REQ2: If the player completes a level and unlocks the next he must have a clear breadcrumb trail back to the title screen via the level screen.

REQ3: If the player presses the exit button, the game will check whether the player wants to quit to system or not and quit the application accordingly.

REQ4: If the game ends or the player completes all levels, the player will be redirected to the title screen.

(sub).3.2 Level Selection

(sub).3.2.1 Description and Priority

The level selection screen is the primary way for the player to choose between levels in case he has exited the game earlier. This game does not support a checkpoint system as gameplay in

each level is timed in between 10-15 minutes.So, a player unlocked level can be accessed through this menu.It is of sorts like a level database, which is essential to the game.

(sub).3.2.2 Stimulus/Response Sequences

Step1: Unlocked levels appear here, there is also an option to return to the main menu.

Step2: The player selects one of the unlocked levels or returns to the title screen.

(sub).3.2.3 Functional Requirements

REQ1: To unlock a level the player must complete the previous level and pass it's in-level cognitive algorithm.

REQ2: When a level is complete, ie: the player has completed all objectives and cleared the cognitive level, he will have the option to view the level menu to play the next level.

REQ3: Only the levels which the player has unlocked can be accessed through the level menu.

(sub).3.3 Pause Menu

(sub).3.3.1 Description and Priority

The player should be able to pause anytime during gameplay, and this screen fulfills that requirement. The pause menu gives the player two options to either restart the level or quit it entirely.

(sub).3.3.2 Stimulus/Response Sequences

Step1: The player hits the associated key which enables the pause menu interface, on the keyboard.

Step2: The level pauses, drawing up the pause menu which prompts the player with two options: “Retry” and “Quit.”.

Step3: The player presses one of the buttons, triggering its respective function.

(sub).3.3.3 Functional Requirements

REQ1: The retry option must return the player to the initial spawn position in the level and reload all the UI values to its initial state.

REQ2: Upon hitting the associated key to trigger Pause Menu, the player must return to the same state he was before entering the menu.

(sub).3.4 Game Over Screen

(sub).3.4.1 Description and Priority

The game over screen must show up when the player is out of level time or has been blocked by the cognitive algorithm.The Game Over Menu gives the player two options to either restart the entire level or to quit the application itself.

(sub).3.4.2 Stimulus/Response Sequences

Step1: The main Timer has run out, or the player is unable to complete the level in time, or the player chooses a wrong logic tile while in the coginive level.

Step2: The player dies, drawing up the Game Over Screen which prompts the player with two options: “Retry” and “Quit.”.

Step3: The player presses one of the buttons, triggering its respective function.

(sub).3.4.3 Functional Requirements

REQ1: The retry option must return the player to the initial spawn position in the level and reload all the UI values to its initial state.

REQ2: The exit option must exit the application completely.

(sub).3.5 Main Timer

(sub).3.5.1 Description and Priority

The main Timer is the most prominent In Game UI of Voyage. It is placed at the top of the game view and gives an estimate of the maximum time the player has to complete a level.

(sub).3.5.2 Stimulus/Response Sequences

Step1: The timer count reduces to Zero minutes.

Step2: The player dies, and the Game Over Screen shows up.

Step3: The player presses one of the options in the screen, triggering it’s respective function.

(sub).3.5.3 Functional Requirements

REQ1: The timer should countdown with respect to world time with no artificial delays.

REQ2: It must constantly update the time left via the in-game UI to provide feedback to the player.

REQ3: As soon as it hits zero, the player gameobject should be destroyed and the Game Over Screen should appear.

(sub).3.6 Codeblock Count

(sub).3.6.1 Description and Priority

The codeblock count Ui element is situated at the top left corner in the In Game UI of Voyage.It gives an estimate of the number of codeblocks a player has picked up during his gameplay.

(sub).3.6.2 Stimulus/Response Sequences

Step1: The codeblock count increases by one, as soon as the player picks up a codeblock

Step2: The codeblock count increments by one at the rate at which the player picks up the codeblock

Step3: The codeblock count stops incrementing once the game objectives are met.

(sub).3.6.3 Functional Requirements

REQ1: The count should keep updating as the player picks up codeblocks.

REQ2:The count should stop updating once the objectives are met.

(sub).3.7 Fragment Count

(sub).3.7.1 Description and Priority

The Fragment count UI element is situated at the top-right corner in the In-Game UI of Voyage.It gives an estimate of the number of Fragments a player has downloaded during his gameplay.

(sub).3.7.2 Stimulus/Response Sequences

Step1: The Fragment count increases by one, as soon as the player interacts with a Fragment.

Step2: The Fragment count increments by one at the rate at which the player interacts with a fragment.

Step3: The Fragment count stops incrementing once the game objectives are met.

(sub).3.7.3 Functional Requirements

REQ1: The count should keep updating as the player interacts with fragments.

REQ2: The count should stop updating once the objectives are met.

(sub).3.8 Health Pickup

(sub).3.8.1 Description and Priority

The Health Pickup increases the health of the player on pickup, it interacts with the player_manager script from UFPS and adds the health to the player on pickup. It is essential as the player is prone to damage within the level by enemyAI and some level structures

(sub).3.8.2 Stimulus/Response Sequences

Step1: The player comes near the health pickup.

Step2: The health of the player gets incremented as soon as he passes through the pickup.

(sub).3.8.3 Functional Requirements

REQ1: It must be continuously updated about the gameObjects surrounding it.

REQ2:It must differentiate between the player and other elements in the game.\

REQ3: It must add a portion of health to the player on collision.

(sub).3.9 Ammo Pickup

(sub).3.9.1 Description and Priority

The Ammo Pickup increases the ammunition of the player's weapon on pickup, it interacts with the weapon_manager script from UFPS and adds the ammo to the player's weapon on pickup. It is essential as the player should be able to deal damage to the enemy UI while he is being attacked.

(sub).3.9.2 Stimulus/Response Sequences

Step1: The player comes near the Ammo pickup.

Step2: The Ammunition of the player's weapon gets incremented as soon as he passes through the pickup.

(sub).3.9.3 Functional Requirements

REQ1: It must be continuously updating the gameObjects surrounding it.

REQ2: It must differentiate between the player and other elements in the game.

REQ3: It must add ammunition to the player's weapon.

(sub).3.10 Codeblock Pickup

(sub).3.10.1 Description and Priority

The Codeblock Pickup increases the Codeblock count and it's reflected in the In-Game UI.It is one of the primary objectives to collect Codeblocks which are scattered throughout the level to power-up the program

(sub).3.10.2 Stimulus/Response Sequences

Step1: The player comes near the Codeblock pickup.

Step2: The Codeblock count increases as the player keeps collecting Codeblock.

Step3: The Codeblock is destroyed with a visual effect as soon as the player collects it.

(sub).3.10.3 Functional Requirements

REQ1: It must be continuously updating the gameObjects surrounding it.

REQ2: It must differentiate between the player and other elements in the game.

REQ3: It must add increment to the value of Codeblock Count in the In-game UI.

(sub).3.11 Floating Fragment UI

(sub).3.11.1 Description and Priority

The FFUI(Floating fragment UI) are UI screens which are scattered across the level. They represent puzzle pieces of the actual memory.There are built in such a way that they will trigger only when the player reaches it. The player is supposed to read the fragments as they are necessary to complete the level.The player can interact with the FFUIs or any UI screens in the level with his weapon.

(sub).3.11.2 Stimulus/Response Sequences

Step1: The player comes near the FFUI. An animation is triggered which displayes the Fragment UI with an option to download it.

Step2: The Player interacts with the button of in the UI and the Fragment Count gets updated

Step3: The Interactive element in the UI gets disabled leaving only the UI for the player reference.

(sub).3.11.3 Functional Requirements

REQ1: It must be continuously updating the gameObjects surrounding it.

REQ2: It must differentiate between the player and other elements in the game.

REQ3: It must add increment to the value of Fragment Count in the In-game UI.

(sub).3.12 Enemy AI

(sub).3.12.1 Description and Priority

The Enemy are Artificial Intelligent systems in the game.Their sole purpose is to locate the player and attack them until he gets spawned back to his initial level checkpoint.There are 3 states in which the EnemyAI performs namely: - The Patrol State, The Attack State and the Chase state. These three states are handled through a coroutine function and it changes depending on the player behaviour and location.The player can either avoid the enemy completely or can destroy them by shooting them through his weapon.The EnemyAi also reflects the age or time of which the level is about.In the prototype level the EnemyAI represents Nazi soldiers.

(sub).3.12.2 Stimulus/Response Sequences

Step1: When the player is in the vicinity of the EnemyAI, the enemy must switch states from patrol to attack and keep attacking the player unless he is killed.

Step2: The Player can destroy the Ai instance by shooting it accurately for a number of times.

Step3:The instance of the AI is removed from the level as the player destroys it.

(sub).3.12.3 Functional Requirements

REQ1: The EnemyAI must be initially in its patrol state where it will consequtively move between Location X to Location Y set by the algorithm.

REQ2:The Enemy must engage with the player once the player is in its line of sight.

(sub).3.13 Upload Codeblocks Interface

(sub).3.13.1 Description and Priority

When the game objectives are met, the player will locate an exit point in the level.On entering the exit point he will find an interface which verifies whether the Objective is true or false.The player is able to interact with it and the interface will show appropriate feedback based on the Codeblock Count.

(sub).3.13.2 Stimulus/Response Sequences

Step1: The Player locates the exit point and the interface and interacts with it.

Step2: The interface provides feedback based on the number of Codeblocks the player has collected. If the Codeblock amount is not met, then it returns an "Insufficient" message.

Step3: if the player has fulfilled the objective, then the interface gives a "Uploaded." Message and blocks any further addition to CodeBlockCount.

(sub).3.13.3 Functional Requirements

REQ1: The Game objectives must be met

REQ2: The Player must locate the Exit Point in the level.

(sub).3.14 Exit Points

(sub).3.14.1 Description and Priority

Green structures mark exit points in the level.Inside each exit Point, the player has to verify whether the game objectives are met. If the objectives are fulfilled the exit point will provide an option to enter a cognitive level where the player will have to test his memory against the algorithm.

(sub).3.14.2 Stimulus/Response Sequences

Step1: The player has met all the objectives of the level

Step2: The player has uploaded Codeblocks and fragments into the respective UI.

Step3: The Cognitive floating UI pops out to allow the player to access the cognitive level.

(sub).3.14.3 Functional Requirements

REQ1: The Game objectives must be met.

(sub).3.15 Upload Fragments Interface

(sub).3.15.1 Description and Priority

When the game objectives are met, the player will locate an exit point in the level.On entering the exit point, he will find an interface which verifies whether the Objective is true or false.The player can interact with it, and the interface will show appropriate feedback based on the Fragment Count.

(sub).3.15.2 Stimulus/Response Sequences

Step1: The Player locates the exit point and the interface and interacts with it.

Step2: The interface provides feedback based on the number of Fragments the player has collected. If the Fragments amount is not met, then it returns an "Insufficient" message.

Step3: if the player has fulfilled the objective, then the interface gives a "Uploaded." Message and blocks any further addition to the FragmentCount.

(sub).3.15.3 Functional Requirements

REQ1: The Game objectives must be met

REQ2: The Player must locate the Exit Point in the level.

(sub).3.16 Automatic Doors, Platforms, etc

(sub).3.16.1 Description and Priority

There are a few automatic doors and trigger platforms scattered across the lavel.These objects trigger an animation when the player is near them.

(sub).3.16.2 Stimulus/Response Sequences

Step1: The Player comes near the object

Step2: An animation is triggered automatically enabling/disabling the players access to his objectives.

(sub).3.16.3 Functional Requirements

REQ1: It must be continuously updating the objects surrounding it.

REQ2: It must differentiate the player from other objects.

REQ3: It must trigger the appropriate animation when the player is around.

(sub).3.17 Automatic Doors, Platforms, etc

(sub).3.17.1 Description and Priority

There are a few automatic doors and trigger platforms scattered across the level.These objects trigger an animation when the player is near them.

(sub).3.17.2 Stimulus/Response Sequences

Step1: The Player comes near the object

Step2: An animation is triggered automatically enabling/disabling the players access to his objectives.

(sub).3.17.3 Functional Requirements

REQ1: It must be continuously updating the objects surrounding it.

REQ2: It must differentiate the player from other objects.

REQ3: It must trigger the appropriate animation when the player is around.

(sub).3.18 In-Game Cognition Level

(sub).3.18.1 Description and Priority

The In-Game Cognition level gets triggered once the player has completed the level objectives and has interacted with the exitUI in the exit point. This level checks the ability of the player to retain information of the fragments which he had collected in the previous levels. It implements this functionality through the use of logic tiles and logic UIs.

(sub).3.18.2 Stimulus/Response Sequences

Step1: The player has interacted with the exitUI and has completed the level objectives.

Step2: A new mini level is loaded which determines the cognitive ability of the player.

(sub).3.18.3 Functional Requirements

REQ: None

(sub).3.18 Logic Tiles and Logic UI

(sub).3.18.1 Description and Priority

The Logic tiles and the logic UI are placed in symmetry to one other.The player reads the information on the logic UI and makes a decision to step on the respective Logic Tile.If he has retained any information of the fragments, the tiles will not collapse.

(sub).3.18.2 Stimulus/Response Sequences

Step1: The player jumps on the tile wth the correct logicUI.

Step2: If the player fails in identifying the correct information, the logic tile collapses and the player dies.He has to restart the level to reach the same point

Step3: If the information of the logic UI is correct, the logic tile does not collapse and the player can successfully reach the Main memory Pickup.

(sub).3.18.3 Functional Requirements

REQ1: The logic tiles must be continuously updating the objects surrounding it.

REQ2: It must differentiate the player from other objects.

REQ3: It must fall apart on the wrong response and stay as it was with the right response.

(sub).3.19 Main Memory Pickup

(sub).3.19.1 Description and Priority

The Main memory Pickup transports the player to the memory interface.The memory has now been unlocked and the player can watch a clip/movie with explains the memory and its importance in Space History. He also has options to go to the level selesct menu or exit the interface altogether.

(sub).3.19.2 Stimulus/Response Sequences

Step1: The player comes near the Main Memory pickup.

Step2: The MMP transports the player to the level memory and starts playing the memory automatically.

(sub).3.19.3 Functional Requirements

REQ1: It must be continuously updating the gameObjects surrounding it.

REQ2: It must differentiate between the player and other elements in the game.

REQ3: It must shift to the Main Memory User Interface as soon as the player collects it.

(sub).3.20 Main Memory User Interface

(sub).3.20.1 Description and Priority

The Main memory User interface allows the player to view the memory related to the particular level.

(sub).3.20.2 Stimulus/Response Sequences

Step1: The Player is within the Main Memory User Interface.The Memory begins playing automatically.

Step2: The Player has options to pause/play memory with a defined input stated in the interface.It also prompts the player with two buttons situated at the bottom right and left corners namely- level Select and Exit.

Step3: The player presses one of the buttons, triggering its respective function.

(sub).3.20.3 Functional Requirements

REQ1:The Interface must load and appear with the memory at all times.

REQ2: If the player presses the exit button, it should exit the application

REQ3: If the player presses the level select button he should return to the level select screen.

Implementation Code Examples:

Below are some key scripts which I coded for "Voyage."These scripts define the main functions and classes which are required to run the game.These appear in the order of System features explained above

1.Loading Scene Coroutines

2.Pause Menu Script

3.Game-UI control script

4.Enemy AI coroutines

5.Codeblock pickup Script

6.Door Animate Script etc.

There are many other scripts which work in co-ordination with one another, but these are some of the key scripts which have been re-applied to other Gameobjects.

1.Loading Scene Coroutines:

Script Name: LoadingBar.cs

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
```

```
public class LoadingBar : MonoBehaviour {

    AsyncOperation ao;
    public GameObject loadingScreenBG;
    public Slider progBar;
    public Text loadingText;

    public bool isFakeLoadingBar = false; //In case of fixed timers
    public float fakeIncrememnt = 0f; //In case of fixed timers
    public float fakeTiming = 0f; //In case of fixed timers
    public int level;




    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        Cursor.visible = true;


    }

    public void LoadLevel01()
    {
        loadingScreenBG.SetActive (true);
        progBar.gameObject.SetActive (true);
        loadingText.gameObject.SetActive (true);
        loadingText.text = "Loading...";

        if (!isFakeLoadingBar)
        {
            StartCoroutine (LoadLevelWithRealProgress ());

        } else
        {
            StartCoroutine (LoadLevelWithFakeProgress ());
        }


    }
```

```
IEnumerator LoadLevelWithRealProgress()
{
   yield return new WaitForSeconds (1);
   ao = SceneManager.LoadSceneAsync (level);
   ao.allowSceneActivation = false;

   while (!ao.isDone)
   {
      progBar.value = ao.progress;
      if (ao.progress == 0.9f)
      {
         progBar.value = 1f;
         loadingText.text = "Press F to continue..";
         if (Input.GetKeyDown (KeyCode.F))
         {
            ao.allowSceneActivation = true;

         }

      }

      Debug.Log (ao.progress);
      yield return null;
   }
```

Explanation: This script sets a coroutine-based on two alternatives at loading. One defines the actual time of load.Depending on the System Specifications this may vary.The other Co-routine defines a fake timer which the developer can set according to his loading screen clip length enabling the player only to load the level once the clip is viewed.

2.Pause Menu Coroutine:

Script Name: PauseMenu.cs

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class PauseMenu : MonoBehaviour {


   //public string levelSelect;
   //public string quit;
   //AsyncOperation ao;
   public bool isPaused;
   public GameObject pauseMenuCanvas;
```

```
    public GameObject Player;
    //public GameObject Camera;

    // Use this for initialization
    // Update is called once per frame
    void Update ()
    {

        if (isPaused) {
            pauseMenuCanvas.SetActive (true);
            Player.SetActive (false);
            //Camera.SetActive (true);
            Cursor.visible = true;
            //Time.timeScale = 0f;
        }
        else
            {
            pauseMenuCanvas.SetActive (false);
            Player.SetActive (true);
            //Camera.SetActive (false);
            Cursor.visible = false;
            //Time.timeScale = 1f;

            }


        if (Input.GetKeyDown (KeyCode.Escape))
        {
            isPaused = !isPaused;

        }
    }

    public void Resume()
    {

        isPaused = false;
    }
    public void Levelquit()
    {
        //ao = SceneManager.LoadSceneAsync (0);
        //ao.allowSceneActivation = true;
        Application.Quit ();
        Debug.Log ("The exit btn was clicked!");

    }
}
```

Explanation: The above code triggers the pauseMenu UI when the "Esc" key is pressed.Initial attempts were made to set the timescale of the Level to zero, but it would cause conflicts with the navmesh and EnemyAI co-routines so this alternative method of setting the Player as false when esc is pressed and enabling the player to true once pause menu is closed works.

3.Game-UI control script

Script Name: UIcontrol.cs

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class UIcontrol : MonoBehaviour {

   public GameObject CodeBlock;
   public GameObject Fragment;
   public GameObject Timer;
   public GameObject MemUI;


   public Text codeBlock;
   public Text fragment;
   public Text timer;


   public float startingTime;
   public static float codeBlockCount = 0f;
   public static float memoryFragmentCount = 0f;

   // Use this for initialization
   void Awake()
   {
      codeBlockCount = 0f;
      memoryFragmentCount = 0f;

   }


   void Start () {

      timer = Timer.GetComponentInChildren<Text>();
      codeBlock = CodeBlock.GetComponentInChildren<Text>();
      fragment = Fragment.GetComponentInChildren<Text>();

   }
```

```
    // Update is called once per frame
    void Update () {

        startingTime -= Time.deltaTime/60f;
        timer.text = "" + Mathf.Round(startingTime) + " Minutes";
        codeBlock.text = "Codeblocks : " + codeBlockCount;
        fragment.text = "Fragments : " + memoryFragmentCount;

        if (TerminalControl.codeblockToggle == true)
        {
            codeBlockCount = 99f;

        }

        if (TerminalControl2.fragmentToggle == true)
        {

            memoryFragmentCount = 99f;

        }

        TerminalCheck ();


    }


    void CheckTimer()
    {

        }

    void TerminalCheck()
    {
        if (codeBlockCount > 40 && memoryFragmentCount > 2 && TerminalControl.codeblockT
oggle == true && TerminalControl2.fragmentToggle == true)
        {
            Debug.Log ("Eureka");
            MemUI.SetActive (true);

        }

    }


}
```

Explaination:

The above code defines variables for Fragment count and Memory Count and updates to check whether a codeblock or a fragment is collected. It then evaluates these variables to a function called Terminal Check which activates the UI which enables the player to access the cognitive level.

4.Enemy AI Script

Script Name::AIEnemy.cs

```
public class AIEnemy : MonoBehaviour
{

   public enum ENEMY_STATE {PATROL, CHASE, ATTACK};

   //private ENEMY_STATE currentState = ENEMY_STATE.PATROL;

   public ENEMY_STATE CurrentState
   {
      get{ return currentState;}
      set
      {
         currentState = value;

         StopAllCoroutines ();

         switch(currentState)
         {
            case ENEMY_STATE.PATROL:
               StartCoroutine (AIPatrol ());
            break;

            case ENEMY_STATE.CHASE:
               StartCoroutine (AIChase ());
            break;

            case ENEMY_STATE.ATTACK:
               StartCoroutine (AIAttack ());
            break;

         }
      }
   }
   //------------------------------
   [SerializeField]
```

```
    private ENEMY_STATE currentState = ENEMY_STATE.PATROL;

    //Reference to the line of Sight Component
    private LineSight ThisLineSight = null;

    //Reference to Nav mesh Agent
    private NavMeshAgent ThisAgent = null;

    //Reference to transform
    private Transform ThisTransform = null;

    //Reference to player health
    public Health PlayerHealth = null;

    //Reference to player transform
    private Transform PlayerTransform = null;

    //Reference to patrol destination
    public Transform PatrolDestination = null;




    //=============================================

    // Use this for initialization
    void Awake ()
    {
        if (this.gameObject != null) {
            ThisLineSight = GetComponent<LineSight> ();
            ThisAgent = GetComponent<NavMeshAgent> ();
            ThisTransform = GetComponent<Transform> ();
            PlayerHealth = GameObject.FindGameObjectWithTag ("Player").GetComponent<Heal
th> ();
            PlayerTransform = GameObject.FindGameObjectWithTag ("Player").transform;
        }

        else
        {
            Destroy (PatrolDestination.gameObject);
            Destroy (this.gameObject);

        }
    }


    void Start()
```

```
{ if (this.gameObject != null)
   {
      //Configure the starting state(co routine)
      CurrentState = ENEMY_STATE.PATROL;
   }
else
{
   Destroy (PatrolDestination.gameObject);
   Destroy (this.gameObject);

   }
}
//================================


public IEnumerator AIPatrol()
{

      while (currentState == ENEMY_STATE.PATROL) {
         //Set Strict search
         ThisLineSight.Sensi = LineSight.SightSensitivity.STRICT;

         //Chase to patrol position
      if (this.gameObject != null) {
         ThisAgent.Resume ();
         ThisAgent.SetDestination (PatrolDestination.position);
      } else {
         Destroy (PatrolDestination.gameObject);
         Destroy (this);
      }

         //wait until path is computed
         while (ThisAgent.pathPending)
            yield return null;

         //If the target is seen start chasing
         if (ThisLineSight.CanSeeTarget) {
            ThisAgent.Stop ();
            CurrentState = ENEMY_STATE.CHASE;
            yield break;

         }

         //wait until nect frame
         yield return null;
      }
```

```
}

public IEnumerator AIChase()
{



    //Loop while chasing
    while (currentState == ENEMY_STATE.CHASE) {
      //Set Loose Search
      ThisLineSight.Sensi = LineSight.SightSensitivity.LOOSE;

      //Chase to last known position
    if (this.gameObject != null) {
      ThisAgent.Resume ();
      ThisAgent.SetDestination (ThisLineSight.LastKnowSighting);
    } else {
      Destroy (PatrolDestination.gameObject);
      Destroy (this);
    }
      //Wait until the path is computed
      while (ThisAgent.pathPending)
        yield return null;

      //Have we reached the destination?
      if (ThisAgent.remainingDistance <= ThisAgent.stoppingDistance) {
        //Stop agent
        ThisAgent.Stop ();

        //Reached Dest but player out of bounds
        if (!ThisLineSight.CanSeeTarget)
          CurrentState = ENEMY_STATE.PATROL;
        else //Reached destination and can see player. Reached attacking distance
        CurrentState = ENEMY_STATE.ATTACK;

        yield break;
      }

      //Wait until next frame
      yield return null;

    }
```

```
}

public IEnumerator AIAttack()
{


      //Loop while chasing and attacking
      while (CurrentState == ENEMY_STATE.ATTACK) {
         //Chase to player position
      if (this.gameObject != null) {
         //ThisAgent.Resume ();
         ThisAgent.SetDestination (PlayerTransform.position);
      } else {
         Destroy (PatrolDestination.gameObject);
         Destroy (this);
      }

         //wait until current path is computed
         while (ThisAgent.pathPending)
            yield return null;

         //Has player run away?
         if (ThisAgent.remainingDistance > ThisAgent.stoppingDistance) {

            //change back to chase
            CurrentState = ENEMY_STATE.CHASE;
            yield break;
         } else {
            //Attack
            Debug.Log ("Attack", gameObject);
         }

         //Wait until next frame
         yield return null;

      }
      yield break;

}

// Update is called once per frame

void Update ()
{
   if (this.gameObject != null) {
      ThisAgent.SetDestination (PatrolDestination.position);
```

```
    }

    else
    {
      Destroy (PatrolDestination.gameObject);
      Destroy (this.gameObject);

    }

  }
}
```

Explanation: This script defines a co-routine based on three states namely Patrol, Chase and Attack. It also uses public functions from LineSight.cs to determine the player is within the Field of View or not.It does this by a sphere collider which acts its range locator from the LineSight script.

5.CodeBlock Pickup Script

Script Name::codepickup.cs

```
using UnityEngine;
using System.Collections;

public class codepickup : MonoBehaviour {

  public Transform codeEffect;


  // Use this for initialization
  void Start () {



  }

  // Update is called once per frame
  void OnTriggerEnter (Collider other)
  {

    if (other.tag == "Player")
    {
      UIcontrol.codeBlockCount++;
      Debug.Log ("Picking up a codeblock!");
      Transform effect = (Transform)Instantiate (codeEffect, transform.position, transform.r
otation);
      Destroy (effect.gameObject, 2);
      Destroy (gameObject);
```

```
    }
   }
}
```

Explanation: This is an example of a collision based script which has been used a lot in the prototype.It adds a value of one as soon as the player collides with it after which it destroys.

6.Door Animate Script

Script Name::DoorAnimateOrange.js

```
#pragma strict

var doorSound : AudioClip;
var doorOpen : GameObject;

function OnTriggerEnter(col:Collider){

  if(col.gameObject.tag == "Player")
    {
      Debug.Log("This", gameObject);
      AudioSource.PlayClipAtPoint(doorSound, transform.position);
      doorOpen.GetComponent.<Animation>().Play("DoorAnim1");


    }
}


function OnTriggerExit(col:Collider){

  if(col.gameObject.tag == "Player")
    {
      AudioSource.PlayClipAtPoint(doorSound, transform.position);
      doorOpen.GetComponent.<Animation>().Play("DoorAnim2");


    }


}
```

Explanation: Initially some Javascript programs were used for Animation and video playback.This pattern is evident throughout the project. This particular script triggers an animation and sound when the player collides with its box collider.

Testing and Evaluation

Testing

The prototype was developed with frequent tests and verification phases. The number of Test cases performed is beyond the scope of this report.However, listed below are some important test cases.

Testing Case 1

Test Case: This test will determine whether the animations related to the enemyAI is working or not.

Test Procedure: Import a rigged model of the Enemy and place it on the scene.Set up the Animation parameters and call them through appropriate scripts(AnimatorSetup.cs, EnemyAnimation.cs)

Expected Result: Animation works perfectly in the environment.

Actual Result: Animation is not working.

Verification: Recheck the blend layers in the character configuration.and also correct vector math algorithms within the EnemyAnimation script.

Result: Animation works now.

Testing Case 2

Test Case: This test will determine whether the switching between the levels are working or not.

Test Procedure: Add relevant loading screens and define functions for each button.Run Scene

Expected Result: Levels are switching perfectly.

Actual Result: Levels are switching perfectly.

Testing Case 3

Test Case: This test will determine whether the pause menu pauses the game time or not.

Test Procedure: Create a pause menu with relevant scripts which triggers on keypress.

Expected Results: The Level Pauses and resumes smoothly.

Actual Result: The Pause functionality works but creates an exception on each frame relating to NavMesh Agent.

Comment: Look for the areas which are causing the error.If it is solved, continue or choose an alternate way to pause the game.

Conditional Test: Write Scripts which can enable/disable the player controller on keypress while reassigning the scene camera controls to another gameObjecr.

Expected Result: Pause Menu works partially with some bugs.

Accuracy: Moderate

Evaluation:

The Evaluation phase for Voyage was partially complete. This was due to time constraints and improper planning in the development stage.

A questionnaire was formulated, and Testers were asked to fill it up once they had completed a playthrough of the level

Questionnaire:

1. Learning Interest: Did this Game make you search for more information on the related historical event while or after Playthrough?
2. How many number of times did you have to restart the level at the “cognitive-firewall” stage?
3. Do you think this method of learning helps in retaining information?
4. Any other game Breaking Bugs you have encountered?
5. Future Suggestions?

Evaluation Participants:

The game is currently in its user testing phase. The users are from a wide age group of acquaintances mostly interested in action first person shooters.A few of them have given feedback.

Hardware and Software Releases:

The prototype was initially launched on the Windows platform. A macosX built was also generated, but it failed to load up due to software limitations.A web-based version was also launched later which could run on most computing platforms.

Criticisms

Gameplay:

Some users felt that the game was not challenging enough. It needs to have a difficulty setting in its Main Interface.

The EnemyAI appeared to be buggy and sometimes the mouse would not register movement with the horizontal or vertical displacement.

Instead of offering an open structure for the player to discover, some felt it would have been better with a map interface which could track objectives.

Learning:

On the learning aspects, most users failed the first time they reached the cognitive level. They had to retry the level again to make it through.

Some felt that the UI was not implemented the right way, and would have been better with AudioVisual narratives rather than static Pictures.

Evaluation Limitations

Players were mainly urged to play the game and gets difficult to evaluate under such circumstances.Most of them lacked motivation, and not with the interest of learning, Hence there was much feedback on gameplay aspects and less on Learning ability.

With a future release with proper marketing through gameplay demos and narratives, this issue could be solved.

Conclusions

The definition of knowledge has changed a lot in the current era.Learning in the modern ages is based more on experience rather than a theoretical understanding of the subject.Games allow users to find and access knowledge, evaluate it in the proper time and context.

In the earlier days, education focused on the acquisition of basic skills and content knowledge like reading, calculations, history, geography and science.However, with the advent of computers, this has drastically changed.Computers can perform complex calculations and algorithms which humans cannot possibly even conceive.

Learning through games is a tested concept and there are many educational games available in the market which incorporates similar themes but most them are either fictional or the message they are portraying is completely out of context.An example could be the popular franchise of Shooters such as Call of Duty or Battlefield. The gameplay elements are refined and apt for the context and age but there are some limitations in this approach. The objectives are mostly redundant with tag and rescue missions or eliminating a threat. Despite the massive historical themes which are force-fed to the user, the implementations of these very themes in the actual game is minimal.

By developing a game in which the users can both have sense of narration and engagement is the purpose of this application. From previous studies, Assassin's Creed has implemented a similar style of engagement but there has also been inaccuracies in depicting history. This prototype on the other hand, tries to keep the historical facts intact and formulate a narrative around it.

Developing this prototype came with its sets of achievements and obstacles.

The Achievements:

After the completion of the prototype, I was confident in working in a game engine environment like Unity. I learnt a lot about Object Oriented Programming and UnityScript ecosystem.I had a chance to learn and implement complex function like IEnumerators, coroutines etc.I could implement some of my former learning in 3d environemnts like Maya and learn its workflow around Unity.

The Obstacles:

Time was the main obstacle as I had to learn everything from scratch.Also, to create an ambitious and playable prototype the system specifications were high and my resources were limited.

Future Prospect

The prototype has recently been released for evaluation, and there are a lot of areas which should be rectified given ample amount of time and effort. The learnability aspects of the prototype is yet to be determined and would require greater User –Testing and researching.

References:

1.Gamestudies.org. (2016). Editorial, Game Studies 0101. [online] Available at: http://www.gamestudies.org/0101/editorial.html

2.Kurt D.Squire, "Video Games and Education : Designing Learning Systems for an Interactive age"Educational technology/March-April 2008

3.Playing Video Games is Good for You, S. (2012). Playing Video Games is Good for You, Study Says. [online] Fox News Latino. Available at: http://latino.foxnews.com/latino/health/2012/03/06/playing-video-games-is-good-for-study-says/

4.Historians.org. (2016). The Assassin's Perspective: Teaching History with Video Games. [online] Available at: https://www.historians.org/publications-and-directories/perspectives-on-history/may-2014/the-assassins-perspective

5.Kurt Squire "From Content to Context:Video games as Designed Experience "https://sashabarab.org/syllabi/games_learning/squire_edres.pdf

6.Cordova, D. I., & Lepper, M. R. (1996). Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. Journal of Educational Psychology, 88, 715–730.

7.Squire, K. D. (2003). Video games in education. International Journal ofIntelligent Games & Simulation, 2(1). Retrieved November 1, 2003, from http://www.scit.wlv.ac.uk/~cm1822/ijkurt.pdf

8.Gee, J. P. (2003). What video games have to teach us about learning and literacy. New York: Palgrave/St. Martin's. Gee, J. P. (2004). Language, learning, and gaming: A critique of traditional schooling. New York: Routledge. Gee, J. P. (2005). Why video games are good for your soul: Pleasure and learning. Melbourne, Australia: Common Ground

9.Clinton, K. A. (2004, April). Embodiment in digital worlds: What being avideogame player has to teach us about learning.Paper presented at the annual meeting of the American Educational Research Association, San Diego.

10.Goodwin, C. (1994). Professional vision. American Anthropologist, 96(3), 606–633.

11.Epress.trincoll.edu. (2016). Pulling Back the Curtain | Web Writing. [online] Available at: http://epress.trincoll.edu/webwriting/chapter/graham/

12.room, m. (2016). Brain Changing Games - Lydia Denworth. [online] Lydia Denworth. Available at: http://lydiadenworth.com/articles/brain-changing-games/

13.Wikipedia. (2016). Unity (game engine). [online] Available at: https://en.wikipedia.org/wiki/Unity_(game_engine)

14.Wikipedia. (2016). Autodesk Maya. [online] Available at: https://en.wikipedia.org/wiki/Autodesk_Maya

15.Wikipedia. (2016). Adobe After Effects. [online] Available at: https://en.wikipedia.org/wiki/Adobe_After_Effects

16.Ux-app.com. (2016). UX-App Documentation. [online] Available at: https://www.ux-app.com/static/docs/index.html

Appendix: Voyage Screen Shots

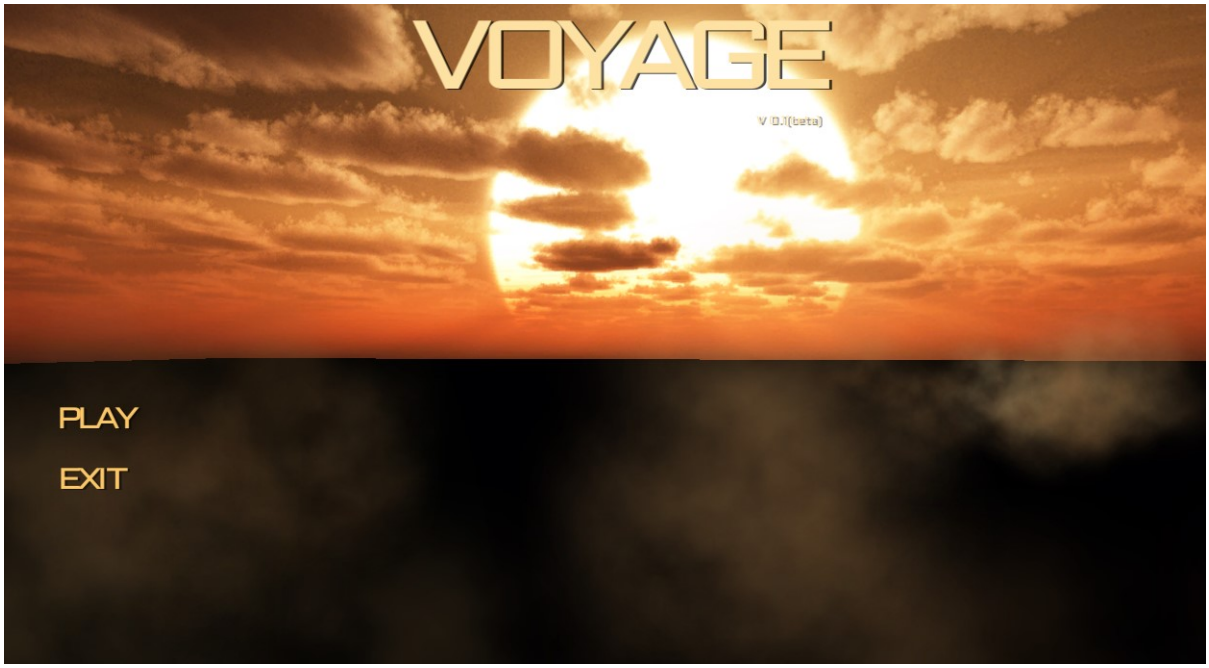


Fig: Title screen of Voyage Prototype.

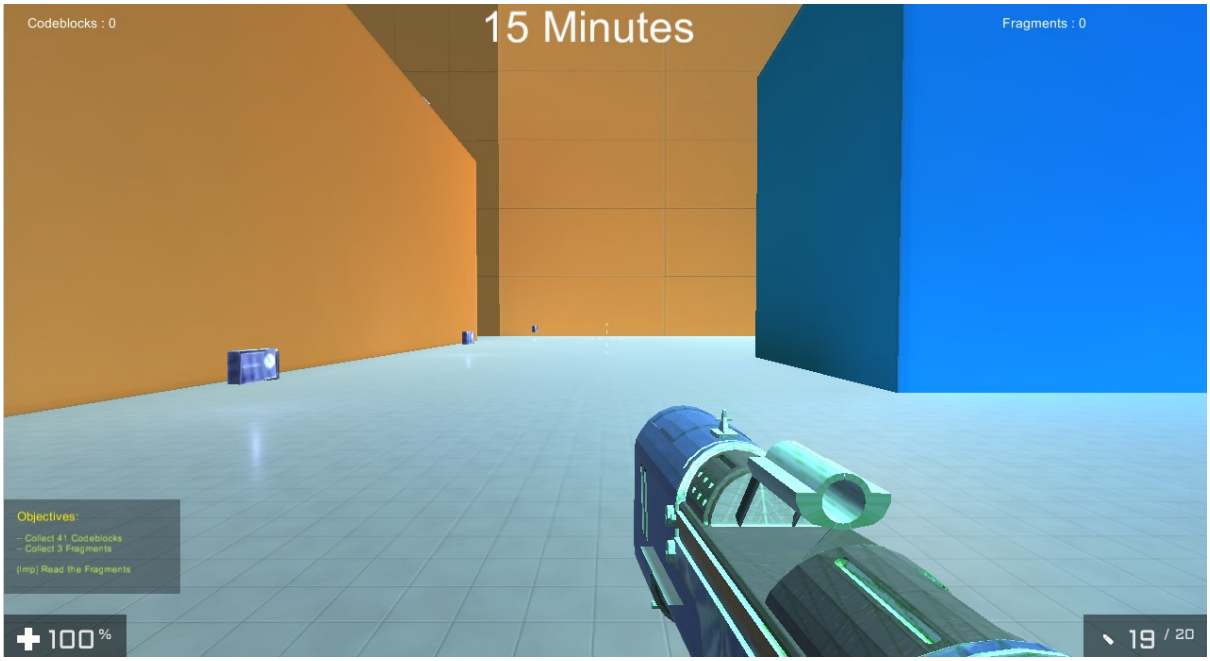


Fig: In-Game UI of Voyage Prototype

Fig: Cognitive Level to access information

Fig: Floating UI in game.

Web Link for prototype:
https://w1569215.users.ecs.westminster.ac.uk/Voyage_game/Voyage%20WebGL%20build.html