DECLARATION

We, the undersigned, declare that this project is our original work and has not been submitted for a degree award in any other institution of higher learning or published anywhere else.

DORWODOE JAMES UE20003415

Signed……. Date: ……

EVANS IGWILO UE20004115

Signed……... Date: ……

OLIVER BOAMAH UE20002915

Signed……. Date…….

This project proposal has been submitted for examination with the approval of our project supervisor

Project Supervisor:

DR. BENJAMIN ASUBAM WEYORI

Signed ……. Date: …….

HEAD OF DEPARTMENT

DR. BENJAMIN ASUBAM WEYORI

Signed ……. Date: …….

DEDICATION

We dedicate this project to the Almighty God for his wisdom and protection. He has made it possible for us to finish our project successfully.

We also dedicate this project to our supervisor Dr. Benjamin Asubam Weyori for his patience, knowledge and directions.

ACKNOWLEDGMENT

We gratefully acknowledge the resourceful guidance, active supervision and constant encouragement of Dr. Benjamin Asubam Weyori who despite his other commitments and busy schedules, made time and helped us in preparing this project report to its present shape. We do convey our sincere Thanks, and gratitude to him. We also thankfully acknowledge the assistance received from my friends, and all those who offered their helping hands to complete this project report directly or indirectly whenever required in preparation of this project.

ABSTRACT

This project document focuses on using gamification to improve learning of English, Science and Mathematics by means of an interactive 3D game called **SMART KIDS.** In order to meet this goal, the group assembled to develop a game targeting mobile devices and windows desktop/laptops since these devices are commonly used by both kids and adults, hence covering a wider target market to help solve the problem at hand.

In today's digital age, it is recognized that children, who are born into a world of technological advancements, are very difficult to educate via conventional methods. As in many other fields, rapidly advancing technology caused certain revisions in the field of education. In today's education architecture, the utilization of digital games is one example of the mentioned revisions.

Adobe Photoshop will be used to produce the graphical user interface to allow the individuals to interact with the game. Blender 3D computer graphics software will be used for creating visual effects, art and 3D models for the game. Unity allows a program to run indefinitely, such as auto-pilot or power plant safety systems, as well as programs that would normally terminate (which here converge to a fixed point). In addition, the use of firmware will permit the hardware to interact with the software of the system. As a result of the project, kids will be able to learn Mathematics, Science and English with ease.

Table of Contents

LIST OF FIGURES

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background

The first video games, developed in the 1960s, were noncommercial. They required mainframe computers to run and were not available to the general public. Commercial game development began in the '70s with the advent of first-generation video game consoles and early home computers like the Apple I. At that time, owing to low costs and low capabilities of computers, a lone programmer could develop a full and complete game. However, in the late '80s and '90s, ever-increasing computer processing power and heightened expectations from gamers made it difficult for a single person to produce a mainstream console or PC game.

Developing software applications is a time-consuming process, and with time-consuming processes come high costs. During the last years, several software development methodologies, often known as agile software development, have become widely used by software developers to

address this issue. Many different development methodologies can be more or less good, depending of the task and application type.

One of the software development methodologies is the evolutionary software method, which, as the name hints, takes on an evolutionary approach to the problem, and allows the project to evolve through different stages of the project. The case study will show how well this evolutionary approach worked on the project on developing a 3D graphic computer game. Some requirements for the computer game were given from the beginning, such as:

- **3D graphics** – The game must contain 3D models, and render these in the game. 3D environments were never a requirement, and platform games with 2D environment could still open up for 3D objects.
- **Impressive result** – The game result must impress whoever plays the game. It should last long, and make the players come back and play it over and over again.
- **Graphical effects** – To achieve an impressive result, modern graphical effects, such as real-time rendered soft shadows, motion blur, and ambient occlusion will be added,

Working with these requirements, The Unity development environment was chosen to develop the proposed 3D gaming software. This decision was made with regard to that it had many in-built tools and provided a good framework for us to get started with the development as fast as possible. Unity is built on C# as a development language.

The requirement for the game to contain 3D graphics introduced an interesting challenge for the project group, since all had none or little experience in 3D modeling. Spending time learning how to model proper 3D models for the game was therefore necessary. During the research to find out what 3D modeling program to use, the need to use different studios to create models which could

later be imported to the proposed game was considered. The complete game would contain models made in both Blender and Photoshop.

The project employed the following tools. Unity to support the framework, and Blender for modeling the 3d graphical components. For sound effects, Fruity Loops or Adobe Audition.

1.2 Problem Statement

In today's digital age, it is recognized that children who are born into a world of technological advancement, are very difficult to educate via conventional methods. The growth of kids is parallel to the future of our country, reflected through the quality of present education system. Formal Education gives kids the chance to acquire knowledge on various fields such as English, literature, history, mathematics, politics, and other numerous subjects. However, there are some drawbacks in the Formal education system with kids:

- Find it Difficult to learn through discussions.
- Easily forget what they read when learning.
- Lack the ability to focus during teaching.
- Respond slowly to questions and problem solving.

1.2 Objectives

1.2.1 Main Objectives

Develop an interactive 3D game to aid school kids in learning the basic SEM (Science, English and Mathematics) subjects.

1.2.2 Specific Objectives

- Enable kids to develop a strategic thinking & problem-solving skill.
- Help kids to learn more about modern technology.
- Use as leisure and entertainment

1.4 Research Question

- What are the principal guidelines for development of a serious game?
- Is it possible to utilize entertainment-based game design in educating?
- What are some of the tools and software that will be used to develop the game?
- Is there an age limit to the game?

1.3 Justification

- Development of the 3D game software will make learning of English, Mathematics and Science more easier and fun, since it will be a type of game with a purpose more focused on learning rather than mere entertainment.

1.5 Scope

This project is confined to the institution "University of Energy and Natural Resources (UENR)" and is limited to the design and implementation of a 3D interactive game for easy learning to enhance kids learning.

1.6 Organization of project.

- **Chapter One**: Introduction- incorporates the problem statement, project objectives, project justification, scope of the work and the organization of the work.

- **Chapter Two**: Literature Review- includes the research and ideologies of existing systems of which my work is emerging from.

- **Chapter Three**: Methodology- involves the developmental processes and flowchart.

- **Chapter Four**: Design and implementation- entails the game architecture, interfaces and also codes in implementing the game.

- **Chapter Five:** Conclusion and Recommendations- summary of what has been done, the outcome of the project and recommendations of the project.

CHAPTER TWO

2.0 Introduction to review

The idea of using games to engage students in the process of active learning is not new. Over the past several years, educators have been increasingly incorporating various games into their teaching curriculum in an effort to create a fun and engaging learning environment for students. Although this can be very challenging and time consuming, interactive, collaborative and competitive games tend to motivate and encourage student participation in the learning process. Over the years, the format for classroom games has changed drastically. There are many more options that incorporate the use of technology and interactivity. Quinn and Iverson argued that students "need to be engaged more and to be put at the center of the learning experience to change from 'passive vessel' to 'active participant'" (as cited in Pannesse & Carlesi, 2007). several games have been conducted with students as a means to review previously taught material and to prepare for tests. It came to noticed that most of the students tend to enjoy

hands-on activities in my courses; however, sometimes when we play games or do activities if they are grasping the content of the material in the process. Some students appear to learn more when they are competing in a game or activity while others seem like they are bored or possibly distracted. As a whole, the feedback received from students regarding the benefits of the review games played has been positive and many students suggest that they play them more often.

One area of significant promise in this regard is a movement toward the use of educational video games as learning tools in schools. In response to this movement, several commercial and custom-made video games have been used in K-12 classrooms across the world to enhance students' learning experience (Wastiau, Kearney, & Van den Berghe, 2009). The 2011 Horizon report suggests that augmented reality and game-based learning will gain widespread use in two to three years (Johnson, Smith, Willis, Levine, & Haywood, 2011). Advocates of game-based learning in higher education cite the ability of digital games to teach and reinforce skills important for future jobs such as collaboration, problem-solving, and communication. While in the past educators have been reluctant to use video games or computer games in the classroom, there is an increasing interest across broad and varied parts of the educational establishment to look at the use of digital games as serious learning and assessment tools. In

2005, the Federation of American Scientists, the Entertainment Software Association, and the National Science Foundation brought together nearly 100 experts to consider ways to develop next generation learning games. They found that many of the skills required for success in games such as thinking, planning, learning, and technical skills are also sought by employers.

Perspectives

There are many explanations as to what defines an "educational game" nowadays. While some games are competitive in nature, others may simply allow students to work together as a class to solve a general problem where no one "wins" or "loses." In "All Play and No Work," MacKenty (2006) states that, "it's the act of problem solving that makes games so engaging devoid of challenge or risk of failure, games really aren't all that much fun". On the contrary, Tom Schrand (2008) discusses the powerful capabilities of interactive multimedia games (or activities) where students work together as a class to categorize information in charts by moving facts so they rest in the appropriate labeled columns. Revisiting these types of games and activities can help with reiterating important information for students. Schaller (2006) states that iteration, or repetition of the process, is critical to "support the learning process by encouraging experimentation, hypothesis testing and synthesis" which are all higher level

thinking skills.

Both formats of gaming activities tend to show learning benefits because of the active learning components that are present in each (MacKenty, 2006, Schrand, 2008). Games that bring out these higher-level thinking skills are becoming more popular, although more research and scientific assessment is necessary to measure their overall effectiveness since they are still relatively new

In this paper, will examine the theoretical and empirical evidence behind five of these claims:

1. Games are built on sound learning principles.

2. Games provide personalized learning opportunities

3. Games provide more engagement for the learner.

4. Games teach 21st century skills.

5. Games provide an environment for authentic and relevant assessment

2.1 RELATED WORK

What Are Digital Games?

For the purposes of this paper, we will use Salen and Zimmerman's (2004) definition of games, which is a "system in which players engage in artificial conflict, defined by rules, that results in a quantifiable outcome". A digital game, then, further refines the definition by requiring the game system to incorporate technology. Simulations, augmented reality, and traditional video games all fall within this definition; however, purely virtual worlds, such as Second Life, would not be games because there is no quantifiable outcome. Elements of "gamification"—the use of game-like mechanisms applied to traditional teaching to increase motivation or engagement (e.g., leader boards, points, badges) or the use of games simply as an extrinsic reward system to increase motivation (e.g., earning game time as a reward for performance) are also not considered games under this definition. While improving motivation and engagement by increasing the fun of learning are indeed important, these types of

approaches are beyond the scope of the paper

Electronic games for education are learning environments that try to increase student motivation by embedding pedagogical activities in highly engaging, game-like interactions. A review of the literature on the effectiveness of games versus traditional classroom instruction can be found. There exist a huge number of electronic educational games, ranging from Mathematics to Language, from Biology to Arts. Nowadays, a new generation of games with a certain instructional component has just risen and tries to spread to a wider audience. Games like "Big Brain Academy" are aimed to engage and entertain the user, but simultaneously improve their skills. "English Training" for Nintendo DS fits the same cluster, but closer to the theme presented. As examples of educational games closely related to the one presented, it can be cited Aqua Spelling [7] for learning orthography or just vocabulary in different languages.

2003 was the year James Paul Gee published What Video Games Have to Teach Us about Learning and Literacy (Gee, 2003), a seminal work cited more than five thousand times in just a decade. At that time, the term "digital native" as coined by game researcher Marc Prensky (2001) was still foreign to the public. Today, a decade after Gee's publication, chances are people are no longer surprised if you use digital games in class. Instead, they'll ask you what you're using the games for. Your Head of Department will probably ask about pedagogies and assessments. What is this game about? What activities would accompany the game? How would you evaluate students' learning? Amazed and amused responses are now replaced by concrete action plans and clarifications. Games are used to help people learn for three major reasons: motivation, content mastery, as well as higher order thinking and social skills.

In the 1980s, MIT professor Thomas Malone (1981) maintained that games motivate players with elements of challenge, fantasy and curiosity. His study contributes to what makes games fun rather than what makes them educational.

Today, the "fun" element of games remains a major point of interest, attracting researchers and practitioners to use games in learning environments. However, it is often uncertain whether players are learning while they are having fun.

Content mastery refers to the acquisition of facts and information generated by experts. Most games designed for content mastery provide opportunities for drill and practice, a dominant learning model in the 20th century. Content mastery games are perhaps the most commonly used and sustainable games in schools due to their close alignment with learning practices in schools.

Higher order thinking and social skills refer to skills used to solve problems, analyze data, synthesize findings and collaborate with others. They are often seen as essential 21st century competencies.

Games that faster (instead of teach) higher order thinking and social skills are often intellectually demanding. They provide challenging contexts for players to experience and develop thinking and social skills. Game scholars like Gee, Kurt Squire, Constance Steinkuehler, David Williamson Shaffer often refer to the learning component in game-based learning as games for higher order thinking and social skills (e.g., Schaffer, Squire, Halverson, & Gee, 2005; Steinkuehler & Duncan, 2008).

Three Views on Game-based Learning

Much like the different reasons for using games for learning, the perspectives on what counts as game-based learning are equally diverse. These perspectives are critical because they pinpoint how game-based learning is conceptualized for learning in schools.

The dominant view conceives game-based learning as a learning approach driven by game technologies (e.g., Gee, 2003; Prensky, 2003). It asks how games, such as a commercial off-the-shelf game, can help young people learn. This view suggests that learning occurs predominantly as a result of game play.

Therefore, learning takes place when players can play a game at their own pace and style, even outside of school. Through self-initiated game play, players develop higher order thinking skills and even social skills without guidance from teachers

The second view perceives game-based learning as a learning approach driven not only by game technologies, but also by pedagogies. Learning does not take place only within a game, but also through the activities designed around a game. Many schools hold this view of game-based learning. Unlike the first view, students (instead of players) often learn to play with guidance from teachers – much like how students learn with textbooks.

The third view regards game-based learning as more of a pedagogical/learning innovation informed by game design principles. This view employs game mechanics and game-design thinking to design learning environments. Role-playing, achievement, competition and reward system are some of the game features often employed to "gamify" the learning contexts such as online communities (e.g., Kapp, 2012).

Game-based Learning in Schools

How are the three perspectives on game-based learning relevant to learning in general, as well as to learning in schools?

Viewing game-based learning as a learning approach driven by game technologies is more suitable for learning initiated by players themselves. When a player plays a complex problem-solving game, such as Civilization, they may develop deep understanding about how a complex system works on their own (Squire, 2004). They learn not only because they are motivated, but also because they can play at their own pace and style.

Viewing game-based learning as both a technological and pedagogical innovation seems like an ideal model for schools as most learning activities involve teacher guidance and the use of technologies. The reality, however, often suggests otherwise.

Games ideal for fostering 21st century competencies are often no longer the same games when they are used in the classroom. The rules of play are changed to fit the schooling context. Players, as students now, are often not allowed to play the game in their own playing styles. The learning processes and outcomes can dramatically change as a result.

Another challenge comes from using a game for what it is not designed for, especially for games with the potential to develop higher order thinking skills. They are often designed as interactive systems that don't "carry" a lot of contents. Even when they do, those contents are essential information for problem solving, instead of canonical facts to be memorized (Gee, 2007; Shaffer, 2006).

While these games are ideal for developing higher order thinking skills, they are unsuitable for content mastery. Teachers usually face the challenge of trying to do both at the same time, given that time is always not on their side. The result is that both teaching objectives are often compromised. Viewing game-based learning as a pedagogical innovation appears to be a more plausible approach, especially when costs, logistics and scalability are major concerns. It is the design principles that make a game a good game and therefore, a good learning experience for a player. Though it is largely unexplored, there is rich potential in employing game design principles to turn the classroom itself into a game for higher order thinking and soft skills.

Games Provide an Environment for Authentic and Relevant Assessment

It is important to note that by definition, games are inherently assessments. Games and traditional assessments share underlying characteristics that provide a means for quantifying knowledge and abilities. The two environments use complimentary technologies that can combine to create more accurate models of student knowledge, skills, and behaviors. For example, games provide opportunities for authentic and appropriate knowledge representation of complex ideas, many of which seem under-represented in traditional assessments (Behrens, Frezzo, Mislevy, Kroopnick, & Wise, 2007). In games, the assessment process occurs as the game engine evaluates players' actions and provides immediate feedback. Players make progress or they don't; they advance to the next level or try again. Assessment occurs naturally in a game. The challenge is assessing the appropriate knowledge, skills, or abilities (Ash, 2011). Methodologies have surfaced as a means for designing games for assessment and quantifying the knowledge and abilities within game environments. Evidence Centered Design (ECD; Mislevy, Almond, & Steinberg, 1998; Rupp et al., 2010) creates a framework for assessment by combining competency, evidence, and task models. This framework defines the attributes being assessed and behaviors that represent such

attributes, and most important, it identifies the activities that connect what is being assessed to what players do within the game (Rupp et al., 2010; Shaffer, Hatfield, Svarovsky, Nash, Nulty, Bagley, Franke, Rupp, Mislevy, 2009; Behrens et al., 2007). This connection between learning, behavior, and setting provides support for the validity of what is being assessed. However, analytic tools are still needed to "score" the observations and update the competency model (i.e., the belief about the player's knowledge or abilities at each point in the game). Koenig, Lee, Iseli, and Wainess (2010) developed a conceptual framework for analyzing the data from interactive games that relies on dynamic Bayesian networks to represent students' real-time actions and decisions. This representation can feed both formative and summative assessments of student performance to provide information about their knowledge, skills, and abilities. Epistemic Network Analysis (ENA) is another tool for translating the elements of ECD as they occur in the game into a knowledge network map. As such, ENA provides snapshots of the player's competency trajectory through the game, which can be continuously quantified, analyzed, and updated to assess the player's development and to inform selection of game task and activities to be presented (Shaffer et al., 2009)

Conclusion

Gaming presents unique opportunities to support the formative process, which is the process by which data about students' knowledge and skills are used to inform subsequent instruction (Heritage, 2010). In order for formative assessments to be useful to instructors and learners, the assessment data must be valid. However, in low stakes assessments students are typically less motivated. Consequently, information gathered about students' knowledge and skills under such circumstances tend to be less valid (Sundre & Wise, 2003; Wise & DeMars, 2003). The increased motivation brought about by games may have the potential to increase the validity of formative

assessments. Delacruz (2011) evaluated games as tools to support formative assessment and examined how varying the level of detail about a game's scoring rules affected learning and performance in mathematics. Her research found that combining elaborated scoring explanation with incentives for accessing game feedback resulted in higher learning gains

CHAPTER THREE

3.0 METHODOLOGY

3.1 Development Process

In a software development project, the resulting product is required to fulfill many different qualities. Examples of such quality requirements are:

robustness, availability, maintainability, dependability and usability. To meet such varying demands, it is important to base the work on a well-prepared strategy.

In software engineering, the term for such a strategy is commonly known as software process, which is built on one or several software processes models.

A software process model is a theoretical philosophy that describes the best way of developing software. Based on one or several models, a software process is formed providing guidance on how to operate. A software process model may also be described as an abstract representation of a software process. The concept of the process model is similar to an abstract java class, which cannot be instantiated, but it can be implemented by another class, thus providing basic guidelines for that other class.

3.1.2 Project Development process

In this project, evolutionary development will be chosen as a keystone for the development process. The decision will mainly be based on the size and the duration of the project. Due limited

time, the best option will be rapid development. Component-based software engineering was easily dismissed since there were very few, or none, components to work with. The waterfall model will be dismissed.

When it came to deciding whether to go with throwaway prototyping or exploratory development, the latter was chosen. Given the circumstances, the best strategy was first to implement the game foundation, and subsequently implement one feature after another, hence using exploratory development.

As indicated earlier, a process is seldom based on one single model, and models are not considered to be mutually exclusive.

Since evolutionary development is a process paradigm that provides little detail on how to work, it would be wise to choose a little more specific model. This model would have to agree with the ideas that evolutionary development represents. There are two major process models which would be suitable; XP (extreme Programming) and agile development methods.

3.1.2.1 Agile Method

Agile is a common name for a number of different software process models. Although the different models share the same idea, that the best way of developing software is throughout incremental development and delivery, they propose different ways to attain this. However, they have one more thing in common, namely a set of principles, which are described below

The users of the software should be closely involved in the development process and contribute by providing new system requirements and evaluate each of the iterations. In this project the developers along with the supervisor acted as the users.

- **Incremental delivery**

  The user specifies what requirements are to be included in what increment.

  People, not process

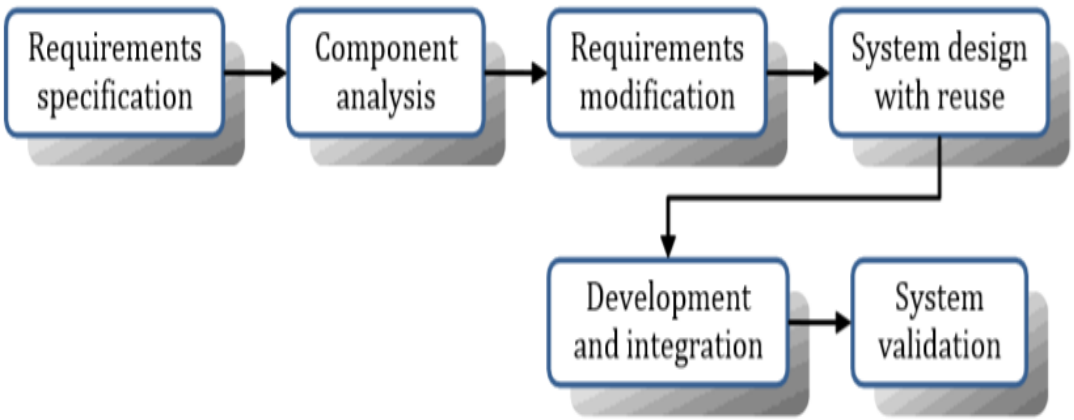
  It is important that the members of the development team are free to develop their own way of working, instead of blindly following what the process specifies.

- **Embrace change**

  The system should be designed in such a way, that it is easy to implement changes or new features.

- **Maintain simplicity**

  Both development process and software should be kept simple.



Agile Development Process fig 1.0

Requirement Specification

We gathered in information's find out the requirements for every average person using a smart phone or a windows laptop or desktop since does are the two platform on which the game will run.

Component Analysis

Both internal and external factors we considered were:

Internal factors involved the tools such as unity 3d, android SDK and the platforms on which the game will operate.

External factor simply involves the individuals who will use the game of which we choose kids and the gadgets which the game will operate on.

Requirement Specification

After our component analysis some few changes were made to the previously set requirement in order to achieve target system design.

System Design with Reuse

Our game development started with only one design but with consultation with our supervisor, he adds more design of which we did n brought it back to him for reviews after use.

Development and Integration

Basically, after our requirement specification to system design was okay, we developed and implemented our game using the various tools

- Adobe Photoshop
- Microsoft paint
- Blender 3D
- Next up Text to Speech
- Audacity Audio Editor
- Microsoft visual C#
- Unity 3D

System Validation:

After development and implementation, we tested the game with our supervisor and some few users to get some feedback and recommendations, which we can implement later on in the game.

3.1.2.2 REASONS FOR USING AGILE METHOD

1. High product quality

In Agile development, testing is integrated during the cycle, which means that there are regular checkups to see that the product is working during the development. This enables the product owner to make changes if needed and the team is aware if there are any issues.

- Defining and elaborating requirements just in time so that the knowledge of the product features is as relevant as possible.
- Incorporating continuous integration and daily testing into the development process, allowing the development team to address issues while they're still fresh.
- Taking advantage of automated testing tools.
- Conducting sprint retrospectives, allowing the scrum team to continuously improve processes and work.
- Completing work using the definition of done: developed, tested, integrated, and documented.
- Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained.

2. Higher customer satisfaction

The product owner is always involved, the progress of development has high visibility and flexibility to change is highly important. This implies engagement and customer satisfaction.

- Demonstrating working functionalities to customers in every sprint review.
- Delivering products to market quicker and more often with every release. The clients get early access to the product during the life cycle.
- Keeping customers involved and engaged throughout projects.

3. Increased project control

- Sprint meetings.
- Transparency.
- Jira usage (visibility of each step of the project for both parties).

4. Reduced risks

- Agile techniques virtually eliminate the chances of absolute project failure.
- Always having a working product, starting with the very first sprint, so that no agile project fails completely.
- Developing in sprints, ensuring a short time between initial project investment and either failing fast or knowing that a product or an approach will work.

- Generating revenue early with self-funding projects, allowing organizations to pay for a project with little up-front expense.
- Agile gives freedom when new changes need to be implemented. They can be implemented at very little cost because of the frequency of new increments that are produced.
- Adaptation to the client's needs and preferences through the development process. Agile commonly uses user stories with business-focused acceptance criteria to define product features. By focusing features on the needs of real customers, each feature incrementally delivers value, not just an IT component. This also provides the opportunity to beta test software after each iteration, gaining valuable feedback early in the project and providing the ability to make changes as needed.

5. Faster ROI

The fact that agile development is iterative means that the features are delivered incrementally, therefore benefits are realized early while the product is in development process.

- Development starts early.
- A functional 'ready to market' product after few iterations.
- First Mover Advantage.
- Long delivery cycles are often a problem for businesses, particularly those in fast-moving markets.

3.2 Game Tools

Adobe Photoshop: - Raster graphics and image editing software developed and manufactured by Adobe Systems Inc; Used in creating the textures for characters / models, creating button images, UI panels.

Microsoft paint: - Simple raster graphics editor that has been included with all versions of Microsoft Windows; used in editing of the UI elements

Blender 3D: - Open source 3D creation suite; Used in creating the 3d virtual Keyboard for user input and the 3d Female character guiding the user

Nextup Text to Speech: - Text-To-Speech (TTS) program; Used to create all spoken audio used during gameplay

Unity 3D: - Cross-platform game engine developed by Unity Technologies; Used to Integrates all game elements / assets, assembling those assets (lighting, audio, physics and animation, interactivity, and gameplay logic) into scenes and environments; and edit, debug and optimize the content for the target platforms.

Microsoft Visual C#: - Programming environment from Microsoft; Used define the logic of game components such as storing / retrieving players data, cash flow logic, level implementation, control UI movement, the character animations.

3.2.1 Game Framework

To save time in the development process, the Unity framework was chosen, to develop the game.

3.2.2 Unity 3D

Unity 3D is a game engine and complete integrated development environment (IDE) with an integrated editor, asset workflow, scene builder, scripting, networking and more. It also has a vast community and forum where any person wanting to know and learn to use Unity can go and have all their questions answered.

3.2.3 Blender and Photoshop

Blender is a professional, free and open-source 3D computer graphics software used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games. Blender's features include 3D modeling, UV unwrapping, texturing, raster graphics editing, rigging and skinning, fluid and smoke simulation, particle simulation, soft body simulation, sculpting, animating, match moving, rendering, motion graphics, video editing and compositing. It also features an integrated game engine for game development. Photoshop can be used extensively for creating of the UI elements for the game.

3.2.3.1 Graphical User Interface (GUI)

Provides an interaction between the gamer and the software (gaming software). This is a major part of the game. If its design is poor it could take a great game and make it average. As this was very important to the success of the project a great deal of time was dedicated to this task.

3.3 Data analysis

Having data of what is happening while the user is playing is key to relating game-play with actual learning, and to move from only theory-based approaches to more data-driven or evidence-based approaches. This in turn can help to contrast the educational approaches and event.

A basic implementation of a Game Learning Analytic system would need to inspect how each player interacts with the game, storing detailed information about the interactions and the changes in the internal game state for further analysis. Such analysis is typically performed in a remote server rather than inside the game, so that data can be aggregated, and analyses tweaked without having to modify game code.

3.4 System flowchart

Show the graphical representation of the flow of data in the **Game application**, and represents the work process of the **Game.**

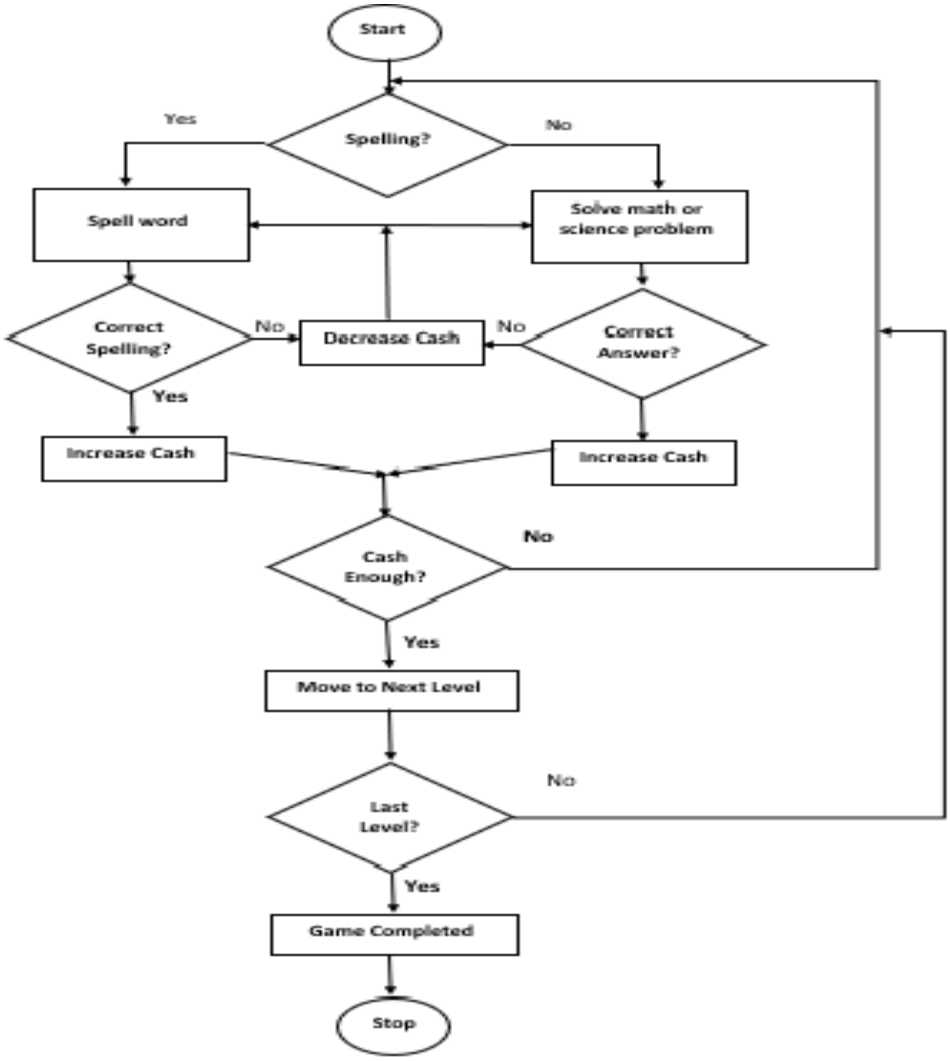


Fig 2.0 Game flow chat

CHAPTER FOUR

4.0 Introduction

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified. Therefore, requirements analysis means to analyze, document, validate and manage software or system requirements. This chapter focuses on the design models and also analysis the models by emphasizing the problem statement and system details such as user interface, game architecture, code for implementing the game software.

4.1 User Requirements

The process of planning, analyzing, designing and implementation of the 3D interactive learning game consisted of steps that included: functional, non -functional and data requirements obtained from the information collected.

Simply put, the difference is that non-functional requirements describe how the system works, while functional requirements describe what the system should do.

4.1.1 Functional Requirement

- Provide an interactive 3D game for easy learning.
- Help the user spell words by providing them with images and their meaning
- Help users solve simple science and mathematics questions
- Able to recall previous user
- Provide user with help functionalities in case of any difficulties during gameplay
- Able to inform user if their answers are wrong or correct.
- Playback audio for better user experience

4.1.2 Non-Functional Requirement

Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability.

Usability:

The game will be used by school kids to improve their form of learning.

Reliability:

The game application will run well, and provide users with a seamless experience.

Support:

The system will run on both android and windows platform without any set backs.

Manageability

The game application does not require any special skills to operate or any expertise's, any kid with a smart phone or a desktop can play the game.

4.2 Unified Language

4.2.1 Functional Modeling

Use-case diagram

Shows the user's interaction with the system that shows the relationship between the user and the various systems. It also identifies the different types of users of a system and the different use cases. In this type of user case an adult interested in the game can play the role of the child.

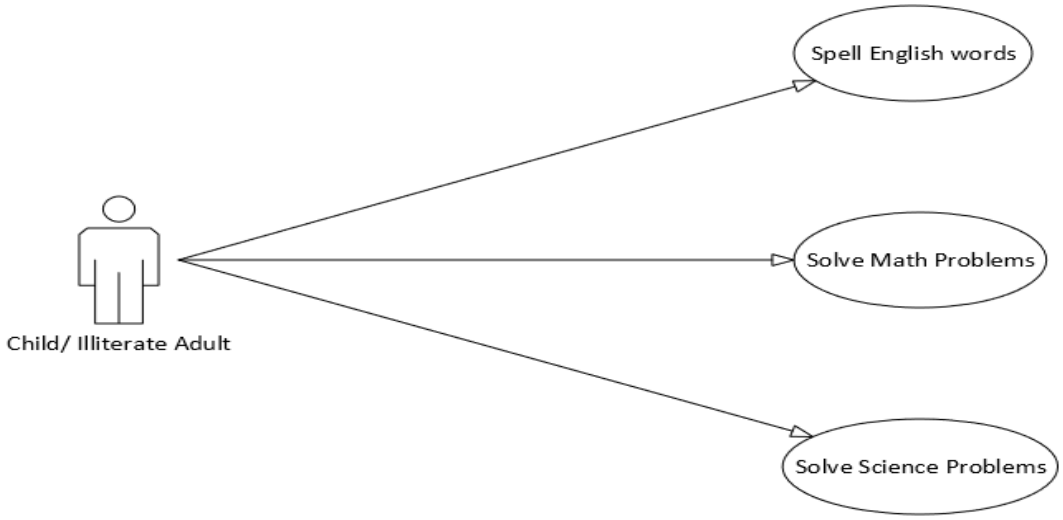


Fig 3.0 User case diagram

4.3 Structure Modeling

Game Architecture

it covers both the theory and practice of game engine software development, bringing together complete coverage of a wide range of system.

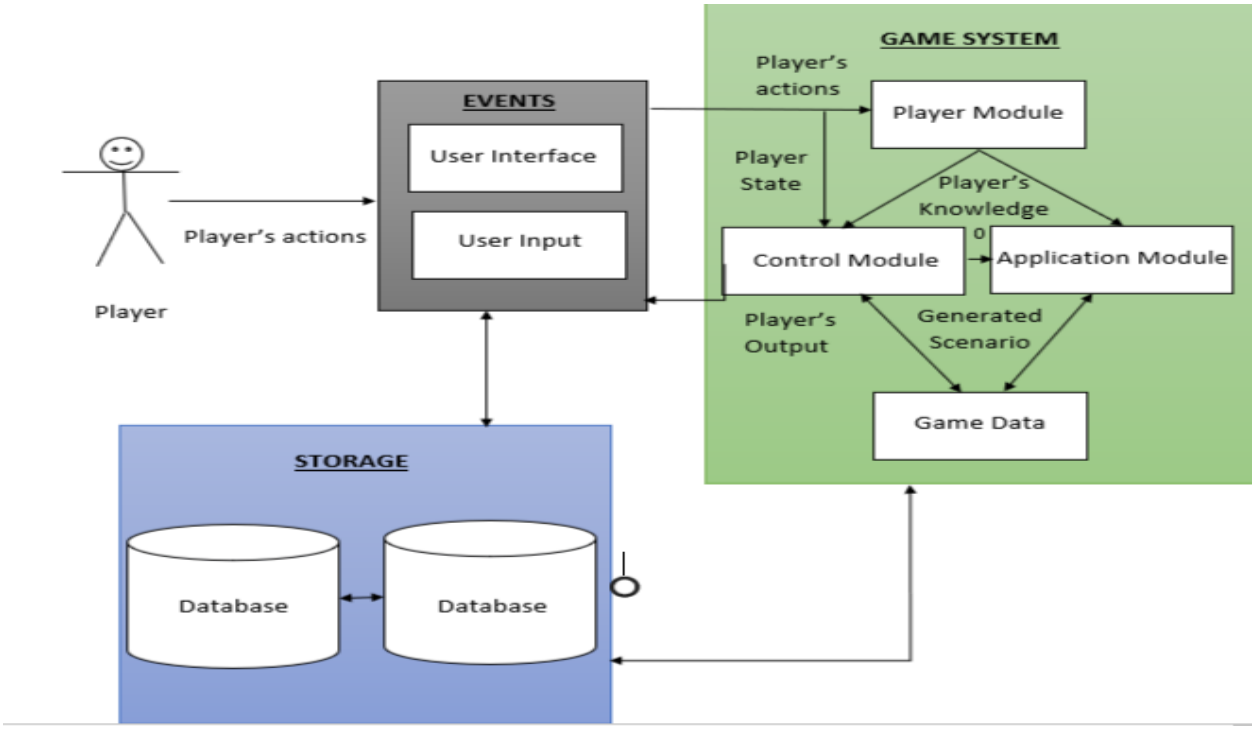


Fig 4.0 Game Architecture

4.4 System Implementation

Implementation is the process that actually yields the lowest-level system elements in the system hierarchy (system breakdown structure). System elements are made, bought, or reused. Production involves the hardware fabrication processes of forming, removing, joining, and finishing, the

software realization processes of coding and testing, or the operational procedures development processes for operators' roles. This part of the project shows the various design, codes and interfaces involved in the game application.

4.5 Coding

The system design is converted into actual working computer codes to implement the game. Below are some of the codes used:

```
using System;

using UnityEngine;

using UnityEngine.UI;



private string[] playerSpellGif;

private string[] playerSpellMeaning;

private bool ShowImageMeaning;

System.Random rand = new System.Random();



int LevelMoney(int currentLevel)

{
```

```
if (currentLevel == 1)

return 500;

else if (currentLevel == 2)

return 1000;

else if (currentLevel == 3)

return 1500;

else if (currentLevel == 4)

return 3000;

else if (currentLevel == 5)

return 6000;

else if (currentLevel == 6)

return 12000;

else return 30000;

}



void GifAnim(string frameName, int totalFrames, float framesPerSecond)

{
```

```
int index = (int)(Time.time * framesPerSecond) % totalFrames + 1;

if (SpellImage.sprite == null || SpellImage.sprite.name != frameName + "_" + (totalFrames - index).ToString("0000") + "_Layer " + index)

SpellImage.sprite = Resources.Load<Sprite>(@"T2S\Spell-0" + ThePlayerInfo.level + "\\" + frameName + "\\" + frameName + "_" + (totalFrames - index).ToString("0000") + "_Layer " + index);

}


void Randomize<T>(T[] items)

{

// For each spot in the array, pick

// a random item to swap into that spot.

for (int i = 0, j = items.Length; i < j - 1; i++)

{

int k = rand.Next(i, j);

T temp = items[i];

items[i] = items[k];

items[k] = temp;
```

```
}

}


void PlayOnce(int speechIndex, int sfxIndex = -1, int levels = -1)

{

if (levels > -1)

{

if (!ReplayQuestion[levels])

{

LevelSample.clip = _SpellMath_Audios;

clipLength = LevelSample.clip.length + Time.time + 0.5f;

playStop = LevelSample.clip.length - 0.4f < 0 ? LevelSample.clip.length :
LevelSample.clip.length - 0.4f;

LevelSample.Play();

ReplayQuestion[levels] = true;

}

}

else if (sfxIndex < 0)
```

```
{
if (!playedSpeeches[speechIndex])
{
AudioSample.clip = Speeches[speechIndex];
clipLength = AudioSample.clip.length + Time.time + 0.5f;
if (speedUp) clipLength *= 0;
playStop = AudioSample.clip.length - 0.4f < 0 ? AudioSample.clip.length : AudioSample.clip.length - 0.4f;
AudioSample.Play();
playedSpeeches[speechIndex] = true;
}
}
else if (!playedSfx[sfxIndex])
{
AudioSample.clip = Sfx[sfxIndex];
clipLength = AudioSample.clip.length + Time.time + 0.5f;
playStop = AudioSample.clip.length - 0.4f < 0 ? AudioSample.clip.length : AudioSample.clip.length - 0.4f;
```

```
AudioSample.Play();

playedSfx[sfxIndex] = true;

}

}



void NextQuestion()

{

for (int i = 0; i < _Current_Length; i++)

{

_All_Letters[_Current_Length - 3][i].text = "_";

_All_Letters[_Current_Length - 3][i].color = Color.black;

_All_Letters[_Current_Length - 3][i].tag = "Untagged";

}



for (int i = rand.Next(0, _Current_Length), j = rand.Next(1, _Current_Length); j > 0; i = rand.Next(0, _Current_Length))

}
```

4.6 User Interface

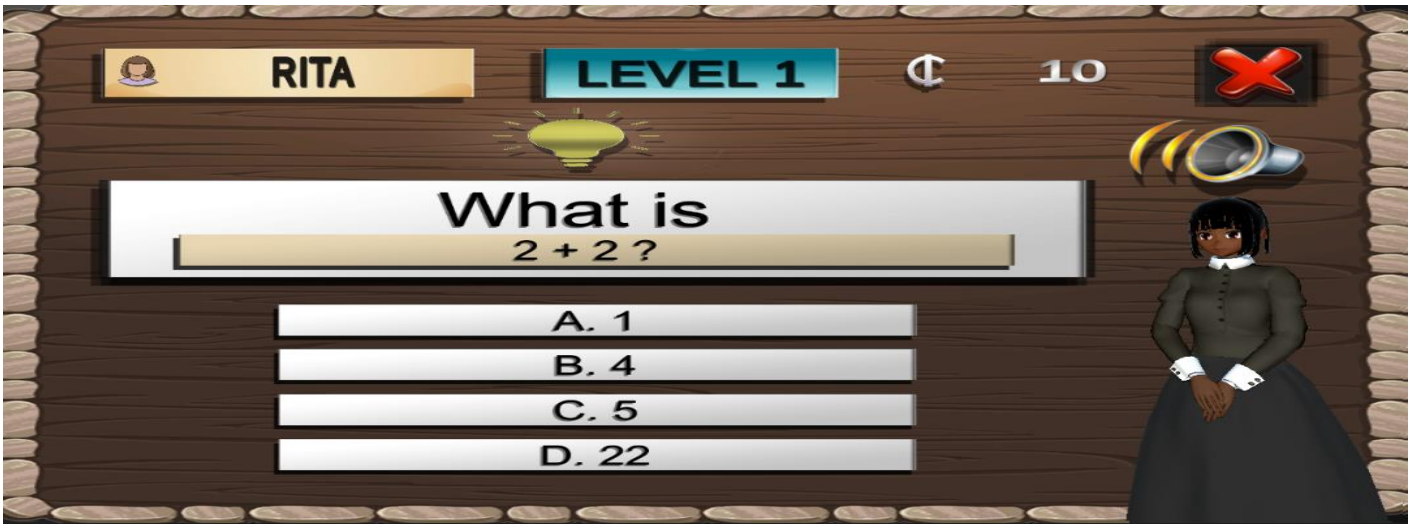
Is everything designed into an information device with which a person may interact. This can include display screens, keyboards, a mouse and the appearance of a desktop. It is also the way through which a user interacts with the game application. The image below shows the basic interface of the game application.

The 3d interactive learning game interface displays the following:

- User name
- Current level
- Amount obtained
- Help function
- Close game function
- Playback audio function
- User keyboard

Fig 5.0 English Words Spelling Interface

Fig 6.0 Mathematics and Science Interface



CHAPTER FIVE

5.1 Introduction

These chapter gives summary of the whole project, its target goals achieved and future recommendations of the project.

5.2 CONCLUSION

Thinking of the game as a part of a bigger educational process is really in the core mind-set that this project wants to promote.

This project aimed as much at using alternative and innovative methods to teach through coding digital games and playing games as part of learning, as at developing the skills of teachers in extending academic goals to understand, support and include the whole child: not only their academic subject skills but also social, emotional and behavioral skills.

The gaming application worked perfectly on both Android and windows platforms which made learning easier and fun for kids using either the smart phone or windows desktop/laptop.

We were also able to achieve our target goal, which was making learning easier for kids.

5.3 Recommendations

For future studies the following ideas can be considered:

- The Gaming Application can be advanced to work on iOS operating system and other operating systems.
- Other educational subjects can be included to the game to make it attractive to a broader scope of users
- System can be improved to have internet access.

5.4 References

Conati, C, and Maclare, H, “Evaluating a Probabilistic Model of Student Affect”, Intelligent Tutoring Systems ITS 2004, LNCS, 3220, 2004.

Randel, J, M, Morris, B, A, Wetzel, C, D, and Whitehill, B, V, “The Effectiveness of Games for Educational Purposes: A Review of Recent Research”, Simulation & Gaming, 23, 1992.

Solet Software, Aqua Spelling, http://www.soletesoftware.com/aquaspelling/es/index.htm

Associaçãoportuguesadesurdos. http://www.apsurdos.org.pt/, 1958.

Ricardo Almeida, Luisa Brito, Pedro Sousa, Marco Capela, Pedro Faria Lopes, and Isabel Alexandre. Gestual life jogo educativo de língua gestual portuguesa. Livro De Atas EPCG - 2014 21o Encontro Português De Computa¸cão Gráfica 21, 2014.

António Andrade Game design document examples. http:// seriousgamesnet.eu/assets/view/238, 06 2013.

Francesco Bellotti, Bill Kapralos, Kiju Lee, Pablo Moreno-Ger, and Riccardo Berta. Assessment in and of serious games: An Overview. Advances in Human Computer Interaction, 2013:1, 2013.

Spires, H. A., Row, J. P., Mott, B. W., & Lester, J. C. (2011). Problem solving and game-based learning. Retrieved from http://www.jmu.edu/assessment/wm_library/Examinee_Motivation.pdf

APPENDIX

```
void NextQuestion()

{

for (int i = 0; i < _Current_Length; i++)

{

_All_Letters[_Current_Length - 3][i].text = "_";

_All_Letters[_Current_Length - 3][i].color = Color.black;

_All_Letters[_Current_Length - 3][i].tag = "Untagged";

}


for (int i = rand.Next(0, _Current_Length), j = rand.Next(1, _Current_Length); j > 0; i = rand.Next(0, _Current_Length))

{

_All_Letters[_Current_Length - 3][i].text = char.ToUpperInvariant(playerSpellQuestions[_Current_Task][i]).ToString();

_All_Letters[_Current_Length - 3][i].color = Color.white;

_All_Letters[_Current_Length - 3][i].tag = "Spell";
```

```
j--;
}

ShowImageMeaning = true;
}

void MeaningImageShow()
{
if (ShowImageMeaning)
{
if (ThePlayerInfo.level > 1)
{
if (_indexMeaning == -1)
if
(char.IsUpper(playerSpellQuestions[_Current_Task][playerSpellQuestions[_Current_Task].Leng
th - 1]))
{
if (!SpellMeaning.gameObject.activeSelf)
```

```
SpellMeaning.gameObject.SetActive(true);

SpellMeaning.text = playerSpellMeaning[Array.IndexOf<string>(playerSpellMeaning, playerSpellQuestions[_Current_Task]) + playerSpellMeaning.Length / 2];

_indexMeaning = 0;

}

if(_indexMeaning == 0)

if (SpellBoard.eulerAngles.y < 180f || _All_Spell_Letter[_Current_Length - 3].transform.localScale.x > 0f || SpellMeaning.transform.localScale.x < 0.07 || SpellMeaning.transform.localScale.y < 0.35)

{

SpellBoard.eulerAngles = Vector3.up * Mathf.Lerp(SpellBoard.eulerAngles.y, 180f + 0.01f, Time.deltaTime * movingVelocity);

_All_Spell_Letter[_Current_Length - 3].transform.localScale = new Vector2(Mathf.Lerp(_All_Spell_Letter[_Current_Length - 3].transform.localScale.x, 0f - 0.01f, Time.deltaTime * movingVelocity), Mathf.Lerp(_All_Spell_Letter[_Current_Length - 3].transform.localScale.y, 0f - 0.01f, Time.deltaTime * movingVelocity));

SpellMeaning.transform.localScale = new Vector2(Mathf.Lerp(SpellMeaning.transform.localScale.x, 0.07f + 0.01f, Time.deltaTime * movingVelocity), Mathf.Lerp(SpellMeaning.transform.localScale.y, 0.35f + 0.01f, Time.deltaTime * movingVelocity));
```

```
}

else _indexMeaning = 1;



if(_indexMeaning == 1)

if (Time.time > clipLength + 3.0f)

if (Math.Abs(SpellBoard.eulerAngles.y) > 0.01f || _All_Spell_Letter[_Current_Length - 3].transform.localScale.x < 0f || SpellMeaning.transform.localScale.x > 0)

{

SpellBoard.eulerAngles = Vector3.up * Mathf.Lerp(SpellBoard.eulerAngles.y, 0.00f, Time.deltaTime * movingVelocity);

_All_Spell_Letter[_Current_Length - 3].transform.localScale = new Vector2(Mathf.Lerp(_All_Spell_Letter[_Current_Length - 3].transform.localScale.x, 1f + 0.01f, Time.deltaTime * movingVelocity), Mathf.Lerp(_All_Spell_Letter[_Current_Length - 3].transform.localScale.y, 1f + 0.01f, Time.deltaTime * movingVelocity));

SpellMeaning.transform.localScale = new Vector2(Mathf.Lerp(SpellMeaning.transform.localScale.x, 0f - 0.01f, Time.deltaTime * movingVelocity), Mathf.Lerp(SpellMeaning.transform.localScale.y, 0f - 0.01f, Time.deltaTime * movingVelocity));

}

else
```

```
{
SpellMeaning.gameObject.SetActive(ShowImageMeaning = (_indexMeaning = -1) > 0);
return;
}
}
if (_indexMeaning == -1)
{
if (!SpellImage.gameObject.activeSelf)
SpellImage.gameObject.SetActive(true);
if (_indexImage == -1)
if (char.IsLower(playerSpellQuestions[_Current_Task][0]))
{
_indexImage = Array.IndexOf<string>(playerSpellGif, playerSpellQuestions[_Current_Task]) +
playerSpellGif.Length / 2;
_indexA = int.Parse(playerSpellGif[_indexImage].Substring(0,
playerSpellGif[_indexImage].IndexOf('-')));
_indexB =
int.Parse(playerSpellGif[_indexImage].Substring(playerSpellGif[_indexImage].IndexOf('-'),
playerSpellGif[_indexImage].Length - playerSpellGif[_indexImage].IndexOf('-')).Trim('-'));
```

```
}

else _indexImage = -2;

if (_indexImage == -2)

SpellImage.sprite = Resources.Load<Sprite>(@"T2S\Spell-0" + ThePlayerInfo.level + "\\" + playerSpellQuestions[_Current_Task]);

else GifAnim(playerSpellQuestions[_Current_Task], _indexA, _indexB);

if (!_indexControl)

if (SpellBoard.eulerAngles.y < 180f || _All_Spell_Letter[_Current_Length - 3].transform.localScale.x > 0f || SpellImage.transform.localScale.x < 1)

{

SpellBoard.eulerAngles = Vector3.up * Mathf.Lerp(SpellBoard.eulerAngles.y, 180f + 0.01f, Time.deltaTime * movingVelocity);

_All_Spell_Letter[_Current_Length - 3].transform.localScale = new Vector2(Mathf.Lerp(_All_Spell_Letter[_Current_Length - 3].transform.localScale.x, 0f - 0.01f, Time.deltaTime * movingVelocity), Mathf.Lerp(_All_Spell_Letter[_Current_Length - 3].transform.localScale.y, 0f - 0.01f, Time.deltaTime * movingVelocity));

SpellImage.transform.localScale = new Vector2(Mathf.Lerp(SpellImage.transform.localScale.x, 1f + 0.01f, Time.deltaTime * movingVelocity), Mathf.Lerp(SpellImage.transform.localScale.y, 1f + 0.01f, Time.deltaTime * movingVelocity));

}
```

else _indexControl = true;

else if (Time.time > clipLength + (_indexA > 15 ? 1.5f : 0.5f))

if (Math.Abs(SpellBoard.eulerAngles.y) > 0.01f || _All_Spell_Letter[_Current_Length - 3].transform.localScale.x < 0f || SpellImage.transform.localScale.x > 0)