Middle East Technical University
Department of Computer Engineering

Initial Design Report for Develop2Learn

Prepared by KIWI

Başak Ece CAN
Emel TOPALOĞLU
Oytun ÖNAL
Hüseyin Cem ÖZTÜRK

Sponsored by

kiwi

12/19/2011

Contents

1. Introduction 3
   1.1. Problem Definition 3
   1.2. Purpose 3
   1.3. Scope 3
   1.4. Overview 4
   1.5. Definitions, Acronyms and Abbreviations 4
   1.6. References 4
2. System Overview 5
3. Design Considerations 7
   3.1. Design Assumptions, Dependencies and Constraints 7
   3.2. Design Goals and Guidelines 8
4. Data Design 9
   4.1. Data Description 9
   4.2. Data Dictionary 10
5. System Architecture 11
   5.1. Architectural Design 11
   5.2. Description of Components 14
   5.3. Design Rationale 33
   5.4. Traceability of requirements 33
6. User Interface Design 34
   6.1. Overview of User Interface 34
   6.2. Screen Images 34
   37
   6.3. Screen Objects and Actions 37
7. Libraries and Tools 38
   7.1. Libraries 38
   7.2. Tools 39
8. Time Planning 40
   8.1. First Term Gantt Chart 40
   40
   8.2. Second Term Gantt Chart 41
9. Conclusion 42

10. Appendix 43
Figure Table 43

1. Introduction

This document describes the initial design strategies and structural properties of the fun-based chemistry learning game which will be developed by Kiwi. It explains the data and interface designs of the project with the system architecture in order to help the developers for better design. This document will also guarantee that the design will correctly implement all the functionalities identified in the SRS document, it will be understandable, efficient, and open to upcoming changes.

1.1. Problem Definition

Currently in Turkey, students have difficulties in learning. This is caused by several reasons. One of the biggest problems is the lack of e-learning resources in schools. Currently MEB is preparing a country-wide project to enhance state of e-learning resources with a project called F@TIH[1]. Within the scope of it, MEB is aiming to distribute tablet computers to 13 million students in Turkey. So, they need plenty of e-learning materials.

Our sponsor W2I and we want to develop a learning material for the students which will be different than the ones already developed. It will be a chemistry learning game which will improve teenagers' knowledge of chemistry as well as improving their critical thinking and creativity skills while they are playing a highly enjoyable game.

1.2. Purpose

This initial design report is aimed to serve as a guideline throughout the development of the project for the developers. It also details what the software requirements are and how they should be implemented.

Its audience consists of developers, which is Kiwi, to give a better understanding of projects, graphics designer to make it clear what should be designed for what purposes, project sponsors to ensure that we meet all the requirements and design project course instructors and teaching assistants to explain them in details what is being developed.

1.3. Scope

The scope of this document contains design patterns of our project. It will be a Unity based chemistry teaching game. Our main target is high-school students. Our aim is to help them with learning chemistry. While having fun, users will improve their chemistry.

1.4. Overview

This document contains initial design of our chemistry teaching game. In the introduction part, we will mostly give the problem definition and the scope of the project. The second part is the overall description of the system including our game scenario. Design constraints are mentioned in the third section. After that, data design and architecture design are given with illustrations. Some sketches and screenshots about the user interface are given in the sixth part. We have given information about Unity packages in section seven. The Gantt charts for the terms are given in the eight section separately.

1.5. Definitions, Acronyms and Abbreviations

D2L:	Develop To Learn
F@TIH:	Fırsatları Artırma Teknolojiyi İyileştirme Hareketi Projesi
MEB:	Milli Eğitim Bakanlığı (Ministry of Education)
W2I:	Words To Inspire

1.6. References

[1] http://fatihprojesi.meb.gov.tr/tr/index.php

[2] http://www.criticalthinking.org/pages/defining-critical-thinking/766

[3] http://www.uwosh.edu/faculty_staff/gutow/VSEPR_TUTORIAL/AX5_right.html

2. System Overview

Our game focuses on the problem solving abilities and encourages students to think creatively. In order to help students embrace the character in the game there will be a story part at the beginning of the game.

<u>The Story at the Beginning</u>

- Two friends are working in the lab. They make chemical experiments with the tubes of elements and compounds, using erlenmeyers, measuring the temperature of the compounds etc.
- Our character feels tired and goes out.
- When he came back he sees that the lab is messy, there are broken glass and shuffled papers everywhere. Moreover, his friend big-brain is not there.
- Then he founds a paper on the floor and realizes that this is a map which big-brain has forgotten.
- He decides to go out and look for him.
- When the goes out he saw that all the people in the neighbourhood is turned into zombies.
- In order to survive and freeze zombies, he takes his liquid nitrogen gun. This starts the game stage 1.

<u>Stage 1</u>

- In stage 1 our character starts with a backpack and a gun. Moreover, many zombies approach the character.
- If the character cannot shoot and allows zombies came near then they will hold the character and start to decrease the health.
- When health decreases to zero, the game starts from the beginning of the stage.
- If character is able to fire his/her gun and to freeze the zombies, they will drop different elements which can be collected to use in the game later.
- These elements will be able to be moved into the backpack by clicking on them.
- When user forgets to take the dropped element he/she cannot continue the game and the shininess around the element will increase to take attention.
- While the character goes forward, he/she needs to jump over the obstacles and freeze zombies.
- After beating some other zombies up, the character came across a liquid on the floor. He cannot jump over, pass through or go around since it is the only entrance into the tunnel.
- In this part, a mouse came and tries to drink some from the liquid. However, it burns the mouse into a skeleton. Here we are going to give the player a hint that shows the liquid is actually acid and he needs to neutralize it.
- The hint guides him to use some litmus paper and see that it turns into red which means it is acidic.
- In order to neutralize he needs to click on the backpack icon on the screen. This opens a new image on the screen showing everything in the backpack. These can be elements or litmus paper.

- There are erlenmeyers in which player can drag the elements and create compounds.
- After creating the compound the erlenmeyer will be poured on the liquid and if they the compound is not a base then a new mouse come and turn into the skeleton or sometimes we expect from the student to use the litmus paper again and see the compound is still an acid.
- Then he needs to try to create a new compound as before and try again.
- He will be able to try until he finishes all the elements (actually the necessary elements, he will have more kind then necessary).
- When the elements finished the stage starts again and he will collect elements again by freezing the zombies.
- When he is able to create the true compound, a sun starts to shine.
- The heat from the sun vaporizes the water in the neutralized substance, salt remains. (Acid + Base = Water + Salt)
- This salt will be moved into the backpack. If user forgets he/she cannot continue and the shininess increases around the salt.
- Moving the salt into the backpack ends stage 1.

<u>Stage 2</u>

- Stage 2 starts into a tunnel whose walls has some clues and tips about Lewis presentation of compounds.
- In this stage the player also needs to beat zombies up. However, this time when they freeze instead of dropping elements they drop stones.
- In order to continue the player should collect these stones by clicking on and sending them into the backpack.
- The health of the player in stage 2 is the same as stage 1. If the player's health decreases to zero, then player needs to start playing stage 2 from the beginning.
- At the end of the tunnel character faces a door on which some element symbols, and holes around them exists.These holes are for electrons they share when elements came together and make a compound.
- Here player needs to select stones in his/her backpack and drag them into the holes on the door.
- If the placement of stones does not fit to the Lewis representation, an earthquake happens and the wrongly-placed stones drop onto the floor.
- If the placement of stones fits to the Lewis representation, the door opens and stage 2 ends.

We are planning to add other stages when we finish implementation of these stages. We have planned our data and system architecture in manner allowing adding new levels to the game.

3. Design Considerations

In this section, special design issues which need to be addressed or resolved before attempting to devise a complete design solution are discussed.

3.1. Design Assumptions, Dependencies and Constraints

In the development of D2L, some specific assumptions should be made considering the software and its use.

3.1.1. Hardware & Software Constraints:

D2L project will be developed to work on multiple platforms. So we will have different hardware and software constraints for these platforms.

For tablet computers (main purpose):

- Operating System: Android 2.0 or higher
- Processor: ARMv7
  (Possible FATIH devices will have Single core 1 GHz or Dual core 800 MHz)
- Memory: 512 MB

For personal computers:

- Operating System: Windows ® 2000 or higher/Mac OS X 10.4 or higher
- Processor: 2 GHz
- Memory: 512 MB

3.1.2. End-User Characteristics:

End-users in our case high school first year students should have a general knowledge of chemistry to be able to play game and pass levels.
Also they should be able to connect to the Internet to share their scores online.

3.1.3. Time Constraints:

The project started in the beginning of 2011-2012 academic year. By end of first semester, first level is aimed to be completed to be able to test it with several students. Before 2012 June, we are planning to complete all coding part to be able to deliver it to all students before the beginning of 2012-2013 academic year.

3.1.4. Graphics Constraints:

Since the game's target audience is high school first year students, graphics should be cool and fancy enough to get their attention. Also, while making them that good device specifications should also be considered.

Since the project's main platform will be the tablet computers that will be distributed by MEB and those devices may not be that fast, our purpose is to keep it as simple as simple as possible. So an implementation of a nice graphics 2D game will be the best solution.

3.2. Design Goals and Guidelines

While designing the D2L project our main purposes are to make it adaptable, sustainable and extensible and to develop critical thinking skills. Since some goals like desirability, ease of use, entertainment factor are obvious, we will not discuss them here.

3.2.1. Extensibility

High school fist year chemistry curriculum is so large that makes it impossible for us to implement game levels for each of these topics within such short time. Whenever a new
topic is desired to be added, it should be implemented easily. Our aim is to make the game in such a way that it should be easy to extend the game by adding more levels with as few changes in the main parts as possible.

3.2.2. Adaptability

We are designing the game for high school freshmen. However, if desired, it should be possible to make scenario and puzzles to adapt other year's topics by using same design concept.

3.2.3. Sustainability

Describe any goals, guidelines, or priorities which dominate or embody the design of the system's software (For example; the KISS principle, emphasis on speed versus memory use, Don't Repeat Yourself principle, portability, or usability, etc.). For each such goal or guideline, unless it is implicitly obvious, describe the desirability.

3.2.4. Promote Critical Thinking

According to Michael Scriven & Richard Paul[2] Critical thinking is the intellectually disciplined process of actively and skillfully conceptualizing, applying, analyzing, synthesizing, and/or evaluating information gathered from, or generated by, observation, experience, reflection, reasoning, or communication, as a guide to belief and action. Critical thinking is essential to effective learning and productive living. This is our main motivation when starting these project. Currently there are lots of chemistry games but none of them promotes critical thinking skills in students. In our game we are planning to develop such skills of students by reasoning, questioning and investigating.

4. Data Design

4.1. Data Description

In Unity development platform in order to ease our job we are planing to create some folders and put our data files into them. Following part is the list of these folders.

<u>Scenes :</u>
The scenes we are using in the game will be hold in this folder. The files have an extension special to Unity.

<u>Libraries :</u>
The packages we loaded form the Unity will be placed in this folder.

<u>Scripts :</u>
The scrips we write are going to be in this folder. Multiple extensions are allowed to be combined in one scene. Javascript and C# scripts can be used.

<u>Game Objects :</u>
Unity allows us to create following game objects:

- Particle System
- Camera
- GUI Text
- GUI Texture
- Directional Light
- Point Light
- Spot Light
- Cube
- Sphere
- Capsule
- Cylinder
- Plane
- Cloth
- Empty Object

We will select some game object among these and add them under game objects folder.

<u>Materials :</u>
The materials we are going to load into objects are going to be hold in this folder. This will include which shader model is going to be used, the color of the object and the texture info of the material. The extension of files is .mat.

<u>Textures :</u>
The 2D graphics we are going to use will be placed in this folder. They will be any picture extension ,but we are planning to use .png files.

Player_Info :
This entity will be stored in player_info.kiwi file which is a special kind of XML file.

4.2. Data Dictionary

While the other data entities are managed by the Unity itself, only the Player_Info entity is used by parts we coded.

This data entity will present the current player's various information such as:

- character name : string
- character gender : char
- the last completed level : int
- achievements : string[]

5. System Architecture

A general description of the D2L game system architecture is presented in the following parts of this section.

5.1. Architectural Design

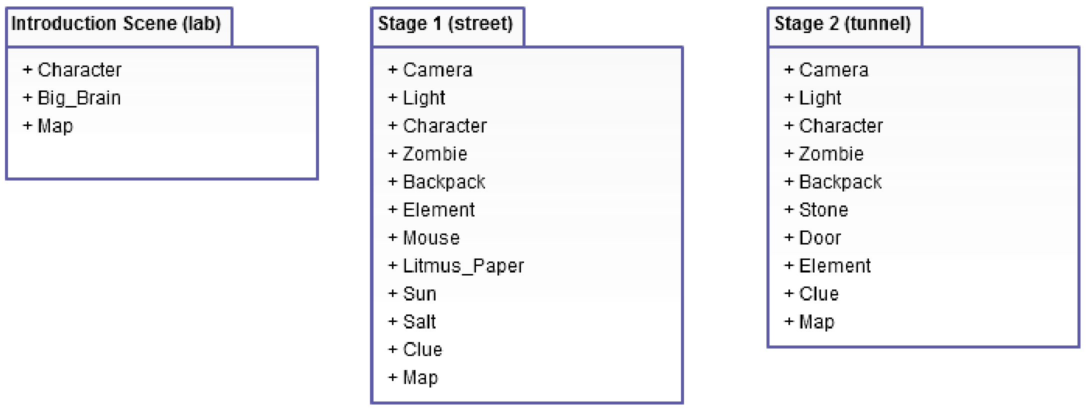


Figure 1: System's architectural design

The game consists of several scenes in which the game_objects are placed and different textures attached to each other. First scene has lab texture and character, big_brain, map objects. In that scene user do not attract with the software, he/she is mostly inactive. Second scene is the first stage of the game. It can be thought as first level. This scene has street texture on the background and camera, light, character, zombie, backpack, element, mouse, litmus paper, sun, salt, clue, map objects. In the second stage the tunnel texture is mapped into the scene. Game objects are: camera, light, character, zombie, backpack, stone, door, element, clue and map. The unity development platform has its own function to pass between the scenes which is LoadLevel script under Application file.

The game objects can be considered as classes in the game and the scripts we are going to write are the functionalities that these objects are going to have. The interactions between these classes are demonstrated in figure 2.

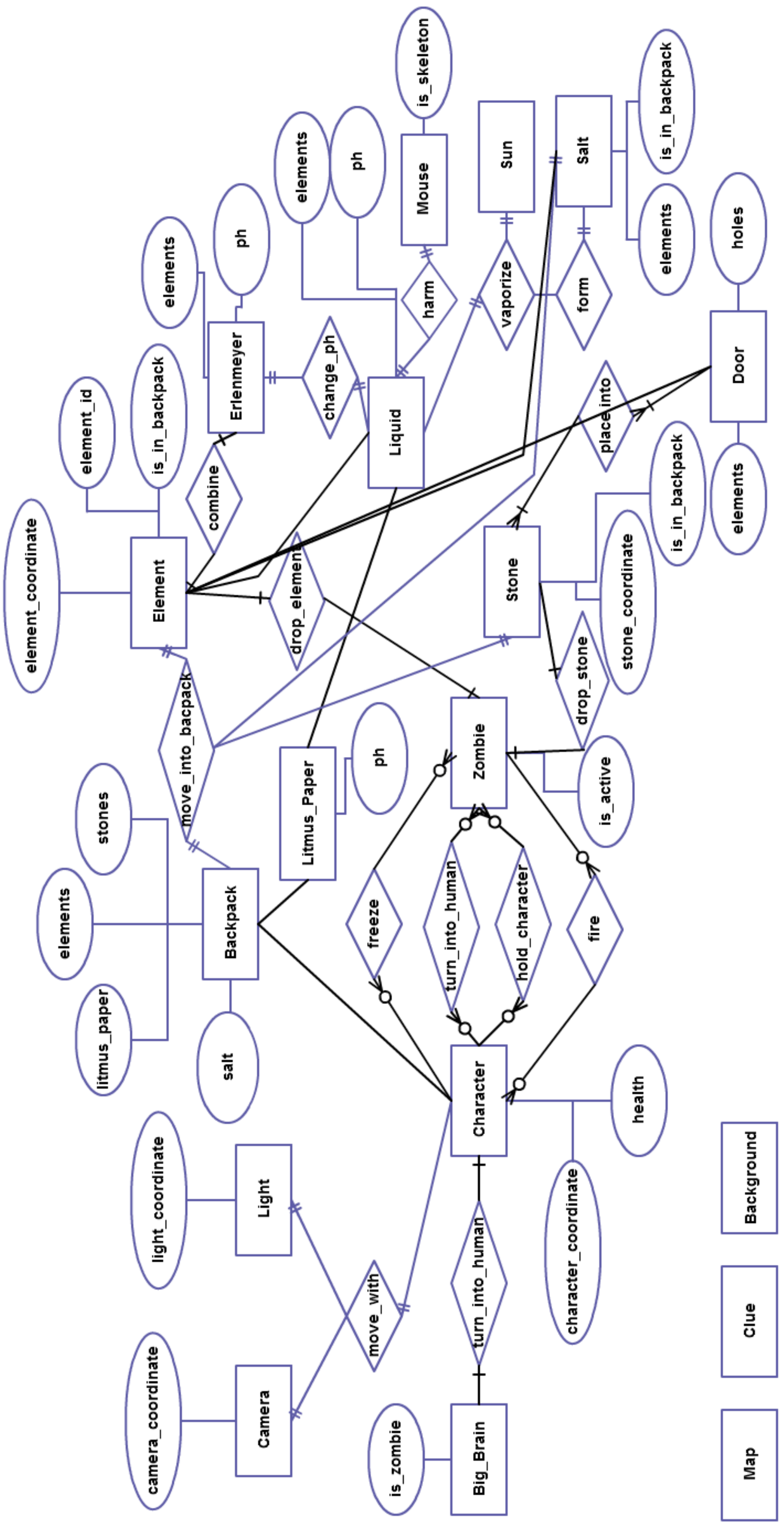


Figure 2: Components interactions

In the game we are planning to use camera and light as followers of the character. For this camera_coordinate and light_coordinate are associated with character_coordinate.

Zombies in the game will be able to decrease character's health by holding him/her and against to that character can freeze and turn the zombies into a human beings by firing. We do not have additional class for human beings. The textures associated with the zombies will be changed after several seconds they became inactive when they are frozen.

When zombies are frozen they can either drop elements or stones near to the character_coordinate according to in which state the player is. In the first stage the elements and in the second stage the stones will be dropped. By clicking the dropped object user will be able to send them into the backpack and then the object will be invisible in the scene. Furthermore, the Boolean, is_in_backpack, will be changed in order to keep this information.

Backpack also includes from the beginning to the end the litmus paper which can be used infinitely. In the game it is going to be used to determine the pH degree of the liquid on the entrance of the tunnel. The danger of the liquid will be illustrated by using a mouse. Which will be going to turn into a skeleton after trying to drink some from this liquid. This is marked with is_skeleton Boolean. The liquid can be neutralized by combining the elements into the Erlenmeyer and change ph degree of the liquid. The element array keeps the element information into the Erlenmeyer. If liquid is neutral, pH=7, the sun vaporizes the water and there will be salt as a remaining substance. The elements make this salt is determined by the elements of the water and the elements in the liquid. This salt can also be taken into the backpack for further use.

The stones dropped are going to use in the Lewis formula which is drawn on the door by filling the holes. For this the Coordinates of the stones and the holes in the door need to be matched.
Map, Clue, and Background classes has no interaction with the other classes their functionalities will be changed with global variables.

5.1.1 About Unity

As we will use Unity in our development process, we would like to explain its features and characteristics which will effect our design decisions.
There is no class, therefore the inheritance logic is absent in the Unity. Instead, game_objects are used as components.

The inputs (keyboard inputs for the desktop PCs and on-screen touches for the tablets) are taken by the Unity, we just need to add event listeners to the game_objects in its update() function. In Unity the objects should have two major functions, awake() and update(). In awake() function the initial characteristics of objects are defined. This function is called once. In update() the necessary changes on objects are defined such as moving the object, changing colors, and interactions with the other objects. This function is called to create frames continuously.

5.2. Description of Components

The components we are going to use are explained in the following section in detail.

5.2.1. Camera

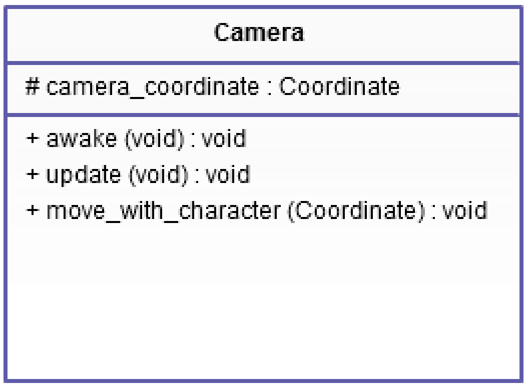

Camera
# camera_coordinate : Coordinate
+ awake (void) : void
+ update (void) : void
+ move_with_character (Coordinate) : void

Figure 3: Camera Component

A Camera is a device through which the player views the world. Unity's camera object already has some predefined functions. We will add the ones above, to these already existing ones.

5.2.1.1. Processing narrative for camera

There will be one main camera in the game which will follow character throughout the game play. When character jumps or crouches it will not move upwards or downwards. It will focus on the character and have it in the center during game with move_with_character function.

5.2.1.2. Camera interface description

Camera will start working when the level starts with awake function. It will periodically call update function to process real time movement of the character, then update what is being displayed on the screen.

5.2.1.3. Camera processing detail

```
User starts level
Awake function is called
While level continues
    Update function is called
    If character moves
        move_with_character function is called with the corresponding character coordinates
If level ends
    Destroy camera object
```

5.2.1.4. Dynamic behavior of camera

Camera interacts with all objects in the scene according to their order of processing. Behavioral diagram of camera and its interactions are given in figure 2.

5.2.2. Light

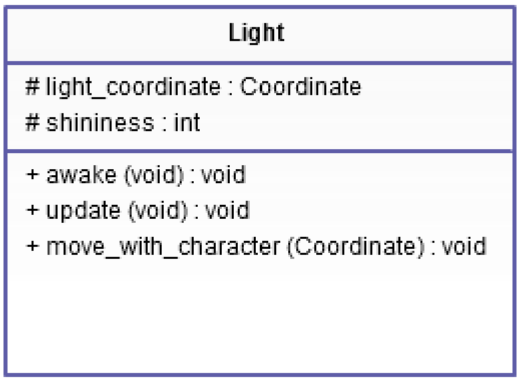

Light
# light_coordinate : Coordinate
# shininess : int
+ awake (void) : void
+ update (void) : void
+ move_with_character (Coordinate) : void

Figure 4: Light Component

We will use lights to illuminate the scenes and objects to create the perfect visual mood. We will use point lights, directional lights and spot lights correspondingly, we will not define any special function for them. The most important light for us is the one that we will attach on character.

5.2.2.1. Processing narrative for component light

In a level there will be various number of lights to illuminate all objects in the game. Most of them will be constant lights. However the light of the character will be dynamic and move with character.

5.2.2.2. Light interface description

There is no special interface defined for lights. User will be able to see their reflections on objects.

5.2.2.3. Light processing detail

5.2.2.3.1. User light object

```
User starts level
Awake function is called
While level continues
    Update function is called
    If character moves
        light's move_with_character function is called with appropriate coordinates
If level ends
    Destroy light object
```

5.2.2.3.2. Other light objects

```
User starts level
Awake function is called
If level ends
    Destroy light object
```

5.2.2.4. Dynamic behavior light

User light objects only interacts with user. Since other lights will be attached to other objects we will not display their interactions in separate diagrams. User light object's interactions are shown in figure 2.

5.2.3. Background

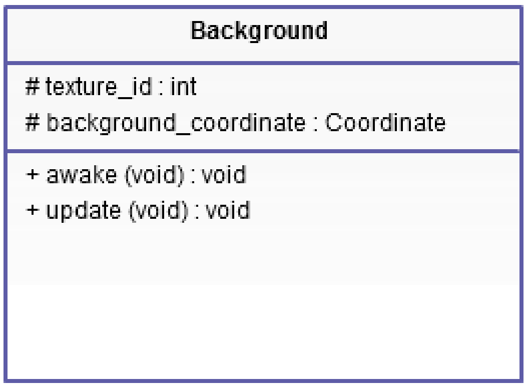


Figure 5: Background Component

Background will be the texture added cube object used to make user understand current atmosphere of the scene. It may be a street, a tunnel, a door or a laboratory depending on the user's progress.

5.2.3.1. Processing narrative for background

Game will start with the laboratory background.In here no action will take place, only story of the game will be told. When main game play starts street background will become active. After solving liquid puzzle tunnel background will be displayed with tips on the walls. At the end of the tunnel a door background will become active to solve a puzzle.

5.2.3.2. Background interface description

Corresponding to the user’s progress in the level required background will be on display.

5.2.3.3. Background processing detail

```
Story begins
    Laboratory background awakened
Game play begins
    Street background is awakened
    While game play is active
        Street background is updated
    User faces with a puzzle
        Street background is destroyed
    If user solves liquid puzzle
        Tunnel background is awakened
    While game play is active
        Tunnel background is updated
    User faces with a puzzle
        Tunnel background is destroyed
        Door background is awakened
    While game play is active
        Door background is updated
Game play ends
    Backgrounds are destroyed
```

5.2.3.4. Dynamic behavior background

Background has no interaction with other classes. Though its place in interaction diagram is still shown in figure 2.

5.2.4. Character

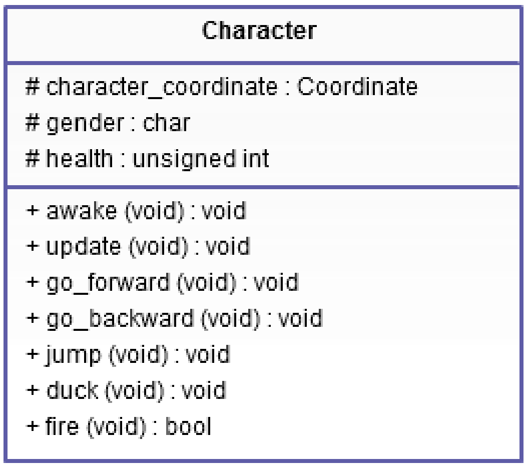


Figure 6: Character Component

Character will be the most active component during game play. It will do some actions like jumping, moving, firing etc.

5.2.4.1. Processing narrative for character

When a level starts character will start to walk in the street or jump, when it faces zombies it will fire to shoot them. What is more before the game play character's gender will be chosen.

5.2.4.2. Character interface description

Character object will have scripts to process keyboard or touch pad input. According to input it will call required functions, and character object will do actions in the screen which is output.

5.2.4.3. Character processing detail

```
User chooses character from menu
    Character is awakened
    Character's gender is set
Game play starts
    while game play is active
        if user clicks move forward button
            go_forward function is called
        if user clicks move backward button
            go_backward function is called
        if user clicks jump button
            jump function is called
        if user clicks fire button
            fire function is called
        if user clicks crouch button
            crouch function is called
        if user is shot
            health is decreased
```

character’s coordinate is updated
character is updated

Game play ends
Character class is deactivated

5.2.4.4. Dynamic behavior character
Character will interact many classes in the game, namely zombie, backpack, light, camera, big brain. Their interaction ways are shown in figure 2.

5.2.5. Map

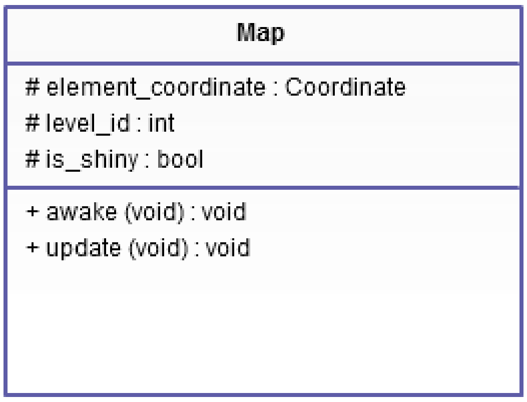


Figure 7: Map Component

Map will be the object to show the gamer his process in the game level. It will also help user to move through the game levels once activated.

5.2.5.1. Processing narrative for map
In the storyline when character returns to laboratory she will find a map shining on the floor. When clicks or touches on it it will be activated. From that on the map will provide user a walk-through of the game. It will have tags like pass the acid liquid, open the Lewis door.

5.2.5.2. Map interface description
In the game play screen maps will be a symbol which user can click on and see what is his place in the game, what to do next.
It will also behave as a level selection screen to go backwards in the game.

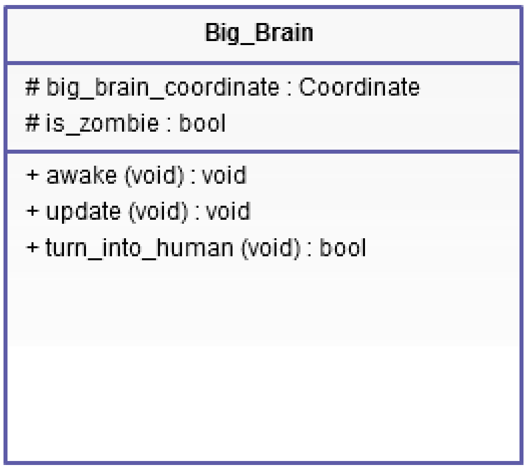
5.2.5.3. Map processing detail
In the story line
When user reaches laboratory map becomes shiny
When user clicks on, map it is activated.
In game play
Map icon is always displayed in the screen

```
        If user clicks on map icon
                Map screen is opened
                If user clicks any of the points in the map
                        User is forwarded to that level
        Map is updated
        If user moves to another level
                level_id is updated
End of game play
        Map is deactivated
```

5.2.5.4. Dynamic behavior map

Map does not intersect with other classes. Though its place is displayed in figure 2 as an independent class.

5.2.6. Big Brain



```
Big_Brain
# big_brain_coordinate : Coordinate
# is_zombie : bool
+ awake (void) : void
+ update (void) : void
+ turn_into_human (void) : bool
```

Figure 8: Big Brain Component

Big brain is the our lab companion who was turned into a zombie accidentally. During the game play character will follow its traces to reach it.
Since our aim will be to return big brain back to human form as well as people of the town. When character reaches it it will prepare a solution or a formula to turn big brain into human form.

5.2.6.1. Processing narrative for big_brain

In the story line when character returns to the lab, he will understand that the big brain has escaped from the lab and turned into a zombie. Character's aim is to find big brain and turn him into human form. When character reaches him, character will prepare a compound to turn in to human.

5.2.6.2. Big_brain interface description

Big process will be only visible in the story line part. User won't be able to control it. It will have some scripts to control its movements.

5.2.6.3. Big_brain processing detail

```
In the story line
    Big brain turns into zombie and escapes form the lab
In game play
    When character reaches big brain it becomes active
    Big brain's coordinates are updated
    If user reaches big brain
        If correct compound is prepared
            Big brain's turn_into_human is called
End of game play
    Big brain is deactivated
```

5.2.6.4. Dynamic behavior big_brain

Only character can interact with big brain and its interaction is displayed in the figure 2.

5.2.7. Backpack

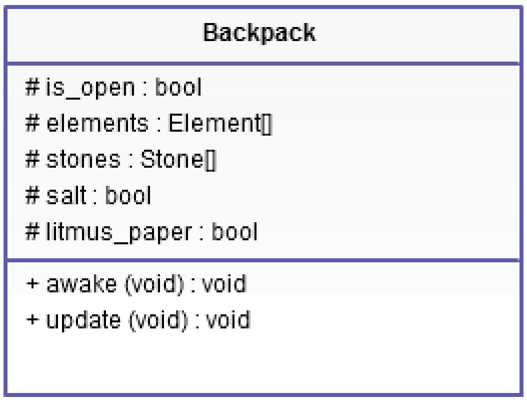


Figure 9: Backpack Component

Backpack will be the object where user keeps his collected items like elements, stones or salt.

5.2.7.1. Processing narrative for backpack

Backpack is activated during game play, when user collects elements, stones etc. they will become reachable from the backpack by clicking on the backpack icon.

5.2.7.2. Backpack interface description

What is inside backpack can be seen by clicking on the icon on game play screen. In such a case click will be our input and display of backpack items will be our output.

5.2.7.3. Backpack processing detail

```
In game play backpack becomes actives
    When user clicks backpack item
        Elements, stones, salt, litmus_paper becomes visible if ant
    Update backpack
```

```
End of game play
    Backpack is deactivated
```

5.2.7.4. Dynamic behavior for backpack

Backpack interacts with stone, salt, litmus_paper and character. Their interactions are displayed in figure 2.

5.2.8. Zombie

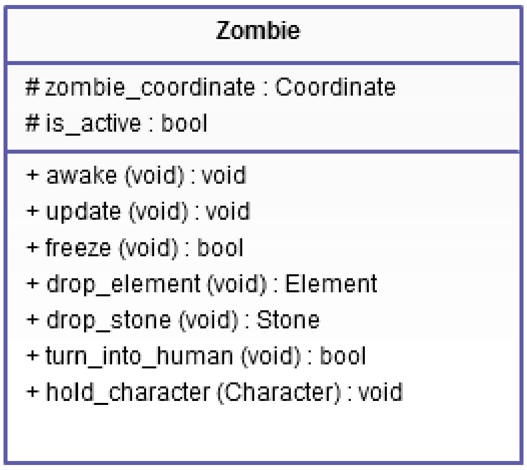

Zombie
# zombie_coordinate : Coordinate
# is_active : bool
+ awake (void) : void
+ update (void) : void
+ freeze (void) : bool
+ drop_element (void) : Element
+ drop_stone (void) : Stone
+ turn_into_human (void) : bool
+ hold_character (Character) : void

Figure 10: Zombie Component

Zombie will be the our opponent in the game which we will fight against to.

5.2.8.1. Processing narrative for zombie

Zombies are normal people that were turned into zombie by big brain. When player completes the game user will be able to turn them into human.
However, during game play user will face them as zombies and will freeze them with nitrogen gun.When they are frozen they will drop elements or stones correspondingly.

5.2.8.2. Zombie interface description

Zombies will become visible without any input, just by reaching corresponding game level.
They will become frozen by shooting.
When they are frozen they will drop elements or stones, which will become visible in the game screen.
If they are turned into human, they will leave zombie form, they will turn into human.

5.2.8.3. Zombie processing detail

```
In game play backpack becomes actives
    When user reaches corresponding part of the level
        Zombie is awakened
        If user freezes zombie
            Zombie drops item(stone or element)
        Else
            Zombie hurts user
            Zombie's hold_user function is called
        Zombie is updated
End of game play
    Zombie's turn_into_human function is called
    Zombie is deactivated
```

5.2.8.4. Dynamic behavior for zombie

Zombie interacts with character,element and stone . And their interactions are shown in figure 2 in detail.

5.2.9. Element

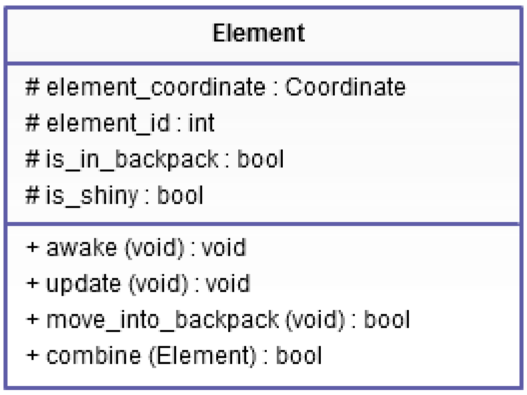


Figure 11: Element Component

Element will be a collectible object that user can collect or combine.

5.2.9.1. Processing narrative for element

Element will be dropped by zombies, and user will collect them by dragging to the backpack. When required they can be combined with same kind.

5.2.9.2. Element interface description

There is no special input makes elements visible. If and only if a zombie is killed they becomes visible by shining on the floor. When they are moved to backpack they cannot be seen on the screen. If the backpack is opened they become visible again to be combined.

5.2.9.3. Element processing detail

```
In game play
    When user kills a zombie
        Element is awakened& becomes shiny
        If element is shiny
            User drags element to backpack by move_into_backpack
            is_in_backpack becomes true
    If user opens backpack
```

```
        If user combines element by dragging it to another element
            combine function is called
End of game play
    Element is deactivated
```

5.2.9.4. Dynamic behavior element

Element interacts with Erlenmeyer, zombie and other elements. Their interactions are displayed in the figure 2.

5.2.10. Mouse

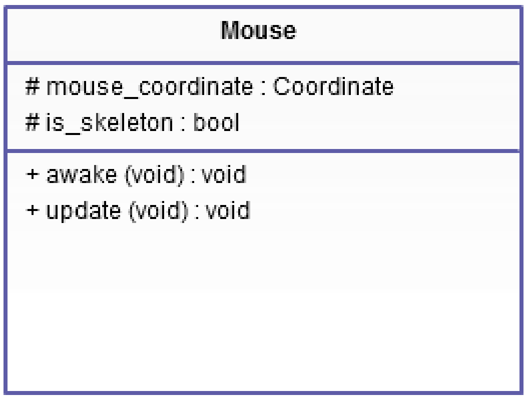


Figure 12: Camera Component

Mouse is the component that will help user the acid level of the liquid barrier.

5.2.10.1. Processing narrative for mouse

When user reaches an acid liquid during game play, he won't be able to move. Instead there will be a mouse object walking to water and drinking it. When the mouse dies, a clue will be displayed to make user understand that that water is acidic and cannot be walked through.

5.2.10.2. Mouse interface description

Mouse will become visible without no input, and will disappear by dying as output.

5.2.10.3. Mouse processing detail

```
In game play
    When user reaches corresponding part
        Mouse is awakened
        Mouse drinks water
        is_skeleton becomes true
    If user fails solving puzzle
        Mouse is awakened
        Mouse drinks water
        is_skeleton becomes true
```

5.2.10.4. Dynamic behavior for mouse

Mouse only interacts with liquid and its details are shown in figure 2.

5.2.11. Liquid

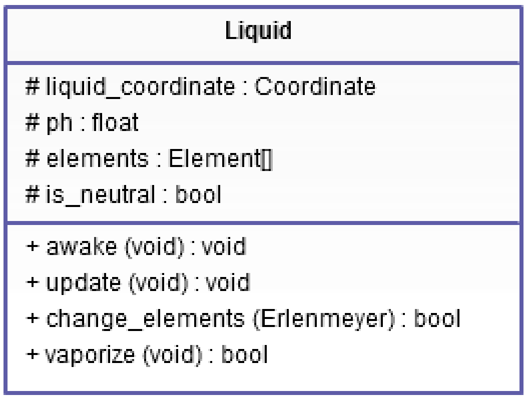


Figure 13: Liquid Component

Liquid is an acidic liquid whose formula will be determined by chemistry teacher. It will be an obstacle on character's way.

5.2.11.1. Processing narrative for liquid

When user reaches corresponding part in the game play, a liquid will be seen. It will behave as a puzzle that user needs to solve to be able to move his journey. This item will has an pH level that can be changed by adding it elements. When it is solved correctly it will be vaporised.

5.2.11.2. Liquid interface description

Liquid will be visible without no input, only when the corresponding level is reached. To be able to pass it user needs to combine elements. And make in neutral. When it is neutralized water is vaporized and user can pass this puzzle by collecting salt.

5.2.11.3. Liquid processing detail

```
In game play
    When user reaches corresponding part
        Liquid is awakened
        Liquid's elements are determined
        Liquid's pH is determined
    If user opens backpack
        If user combines correct element to neturalise it
            is_neutral becomes true
    If is_neutral true
        Water is vaporized
    Liquid is updated
```

5.2.11.4. Dynamic behavior liquid

Liquid interacts with Erlenmeyer, litmus paper, element and sun. Their interactions and relation types are shown in figure 2.

5.2.12. Litmus Paper

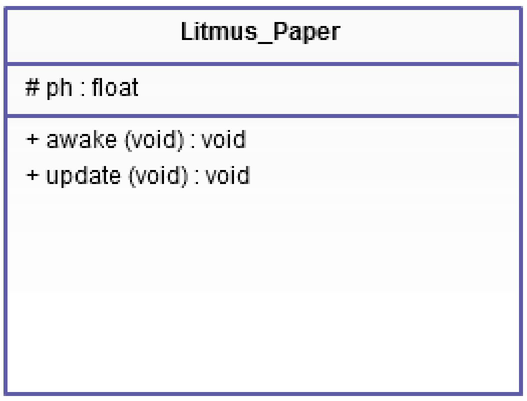


Figure 14: Litmus Paper Component

Litmus paper will be used to determine the pH of the liquid.

5.2.12.1. Processing narrative for litmus paper

When user reaches, liquid she will be able to determine its pH by dragging a litmus paper to it.

5.2.12.2. Litmus paper interface description

Litmus paper becomes visible when backpack is opened. It can be dragged to liquid and used to determine pH of liquid.

5.2.12.3. Litmus paper processing detail

```
In game play
    When sees liquid
        If user opens backpack
        Litmus paper is activated
        User drags it onto liquid and pH is determined
```

5.2.12.4. Dynamic behavior litmus paper

Litmus paper only interacts with liquid and its interactions are shown in figure 2.

5.2.13. Erlenmeyer

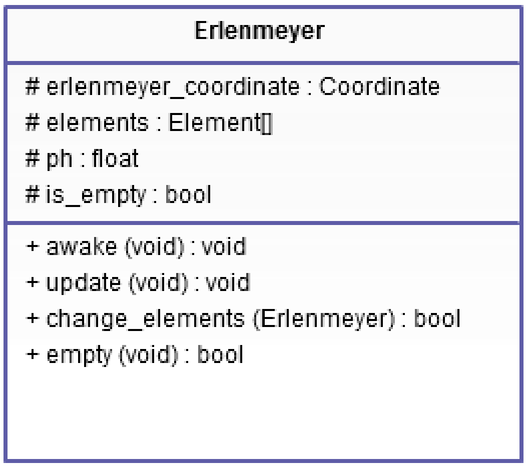


Figure 15: Erlenmeyer Component

Erlenmeyer will be the game object where user can combine elements into it.

5.2.13.1. Processing narrative for Erlenmeyer

Erlenmeyer will be used to combine elements in the required parts of the game play such as acidic liquid puzzle. It will hold the pH of objects whose are combined within it. If a wrong combination is prepared user will be able to empty it and prepare a new compound in it.

5.2.13.2. Erlenmeyer interface description

It will become visible when backpack item is clicked with the shape of an Erlenmeyer. When elements dragged into it it will look like a full item and when it is empty it will look as it is. When backpack is closed it won't be visible.

5.2.13.3. Erlenmeyer processing detail

```
In game play
    Erlenmeyer is activated
    Erlenmeyer is set to be empty
    If user opens backpack
        If user drags elements to Erlenmeyer
            Erlenmeyer's elements are changed
            pH is changed
        If compound is wrong
            Erlenmeyer is emptied
            Erlenmeyer's elements are set to null
        Erlenmeyer is updated
End of game play
    Erlenmeyer is deactivated
```

5.2.13.4. Dynamic behavior Erlenmeyer

Erlenmeyer interacts with elements and liquid. Their interactions are shown in the figure 2.

5.2.14. Sun

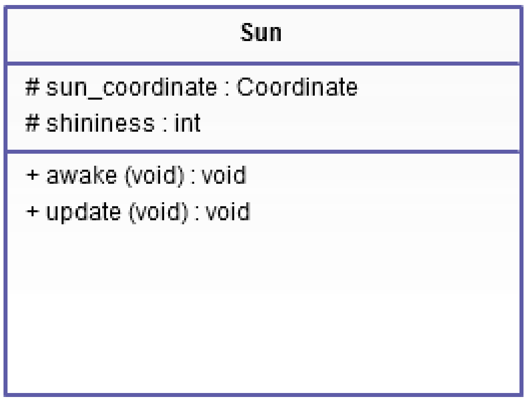


Figure 16: Sun Component

Sun will be a game object which vaporizes the water.

5.2.14.1. Processing narrative for sun

Sun will occur after the correct compound is made in erlenmeyer and poured into the liquid and neutralized it. Its heat vaporizes the water formed with this process.

5.2.14.2. Sun interface description

Sun becomes visible when the acid-base puzzle is solved and invisible when salt is separated.

5.2.14.3. Sun processing detail

```
In game play
    If user pours the correct compound into liquid
        Sun rises
        Water is vaporized
    If salt is formed
        Sun disappears
```

5.2.14.4. Dynamic behavior sun

Sun interacts with liquid. Its interaction are shown in the figure 2.

5.2.15. Salt

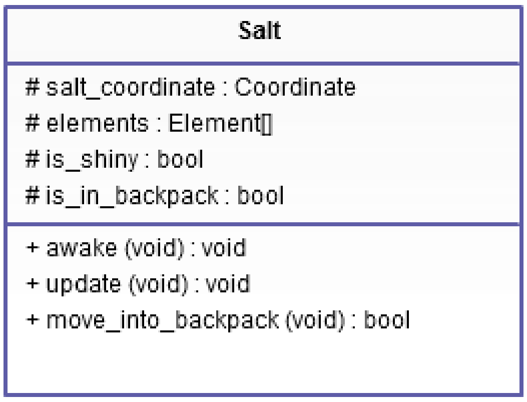


Figure 17: Salt Component

Salt is the object that will be collected when water in the liquid is vaporised by the sun.
It will also take place in the backpack and will be used in the following levels.

5.2.15.1. Processing narrative for salt

When liquid puzzle is solved, sun shines, as it vaporizes all water salt becomes visible and shiny. It should be put into the backpack to be able to continue the level progress.

5.2.15.2. Salt interface description

Salt is activated by the sun shine. When it is moved to backpack it will lose its visibility. From that on it will be only an item in the backpack.

5.2.15.3. Salt processing detail

```
In game play
    If sun shines water vaporizes
        Salt becomes active
        Salt's is_shiny is set to true
    If user drags it into backpack
        move_into_backpack function is called
        Salt's is_in_backpack becomes true
    Salt is updated
End of game play
    Salt is deactivated
```

5.2.15.4. Dynamic behavior for salt

Salt has interactions with backpack, sun and liquid. Their interactions are displayed in figure 2.

5.2.16. Clue

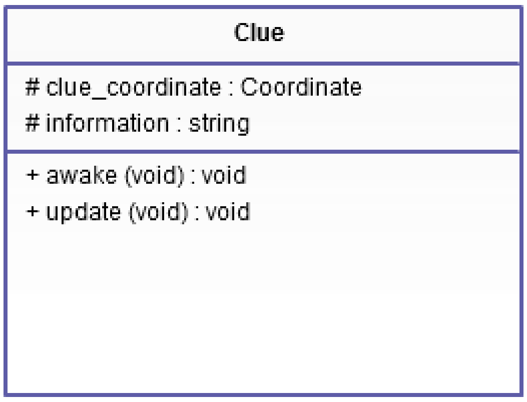


Figure 18: Clue Component

Clues in the game play will help user to move forward in the game with small tips.

5.2.16.1. Processing narrative for clue

When user reaches previously determined parts of the game play, clue object will shown there and will be collectible. As they are collected their tips will appear in the screen.

5.2.16.2. Clue interface description

Clue has no special input to make it active. Just reaching the required part of the level will be enough. If user clicks on it it will show the tip on the screen and they disappear after a while without no special function.

5.2.16.3. Clue processing detail

```
In game play
        User reaches corresponding level part
        Clue is activated
        If user clicks on it
                Clue’s information string is displayed on the screen
End of game play
        All clues are deactivated
```

5.2.16.4. Dynamic behavior for clue

Clue is an independent class in the game, though its place is still shown in figure 09283409238490234.

5.2.17. Stone

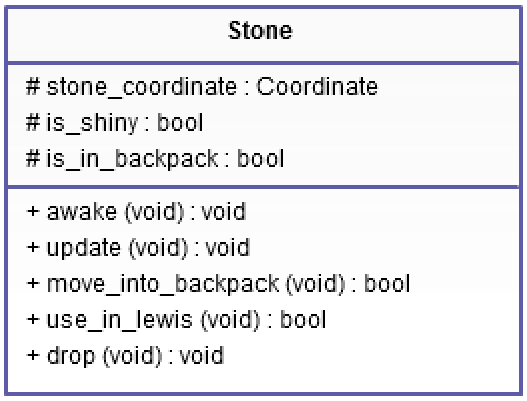


Figure 19: Stone Component

Stone will be a collectible object that user can collect or combine.

5.2.17.1. Processing narrative for stone

Stone will be dropped by zombies, and user will collect them by dragging to the backpack. In addition, they will be placed into the holes on the door. When user make a mistake the earthquake make them drop into the floor. Therefore, player should drag them to the backpack and use them again.

5.2.17.2. Stone interface description

Stones become visible and start to shine when zombie is frozen. When they are moved to backpack they cannot be seen on the screen. If the backpack is opened they become visible again to be dragged to the holes on the door.

5.2.17.3. Stone processing detail

```
In game play
    When user kills a zombie or earthquake happens
        Stone is awakened& becomes shiny
        If stone is shiny
            User drags stone to backpack by move_into_backpack
            is_in_backpack becomes true
    If user opens backpack
        If user drags stone to the hole of the door
            use_in_lewis function is called
End of game play
    Stone is deactivated
```

5.2.17.4. Dynamic behavior for stone

Stone has interactions with zombie, door and element. Their interactions are displayed in figure 2.

5.2.18. Door

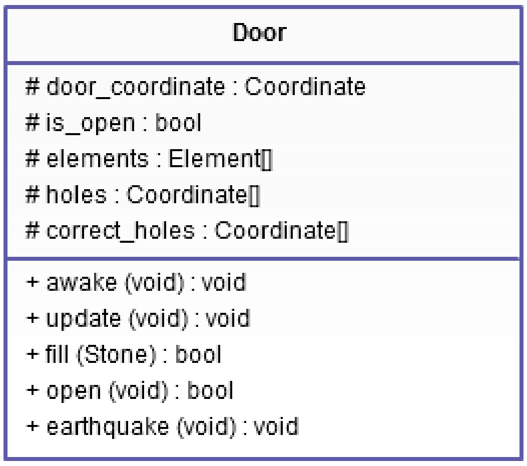


Figure 20: Door Component

Door will be the object where Lewis puzzle is displayed. If it is solved correctly it will allow user to pass another level of the game which will be designed later. A sample filled version of door puzzle is shown in figure 25.

5.2.18.1. Processing narrative for door

Door will be active when user reaches it. It will have holes on itself to be filled with stones in order to display correct Lewis formula. If stones are put correctly it will be opened, otherwise there will be an earthquake and all stones drop, holes will be empty again.

5.2.18.2. Door interface description

It will be visible without no special input, and user will be able to move stones to it by dragging onto wall. Each time a stone is dragged door is updated. When is is filled correctly it will disappear.

5.2.18.3. Door processing detail

```
In game play
    When user reaches corresponding part
        Door’s elements are set
        Door’s is open is set to be false
        Door is activate
    If user drags a stone to any hole
        corresponding hole is filled
    If all holes are filled
        If correct_holes are true
            Door’s is open set to be true
            Open is called to make user to move onto another level
        Else
            earthquake is called
            elements are changed
```

holes are emptied
Door is updated
End of game play
Door is deactivated

5.2.18.4. Dynamic behavior door

Present a description of the interaction of the classes. Present a sequence diagram for each use case the component realizes.

5.3. Design Rationale

We are forced to use Unity while developing our game ,so we chose this design to be parallel to the Unity Development Tool.

5.4. Traceability of requirements

When we look at Unity deeply we have seen that most of the requirements we have determined do not match with the Unity. Nearly all the requirements are in the Unity packages. We do not need to make a design for them. Therefore, in this document we have focused and explained mostly what we will write.

6. User Interface Design

6.1. Overview of User Interface

For our chemistry teaching game, graphical user interface which will provide users an initiative experience will be implemented. Our main purpose, while designing GUI, is to increase playability as much as possible. Moreover GUI should be user friendly. User plays the game via control button implemented on GUI. There will be direction controller which enables user to make their character move. Also attack button and jump button will be provided. In addition hints and tips will be available through GUI.

6.2. Screen Images

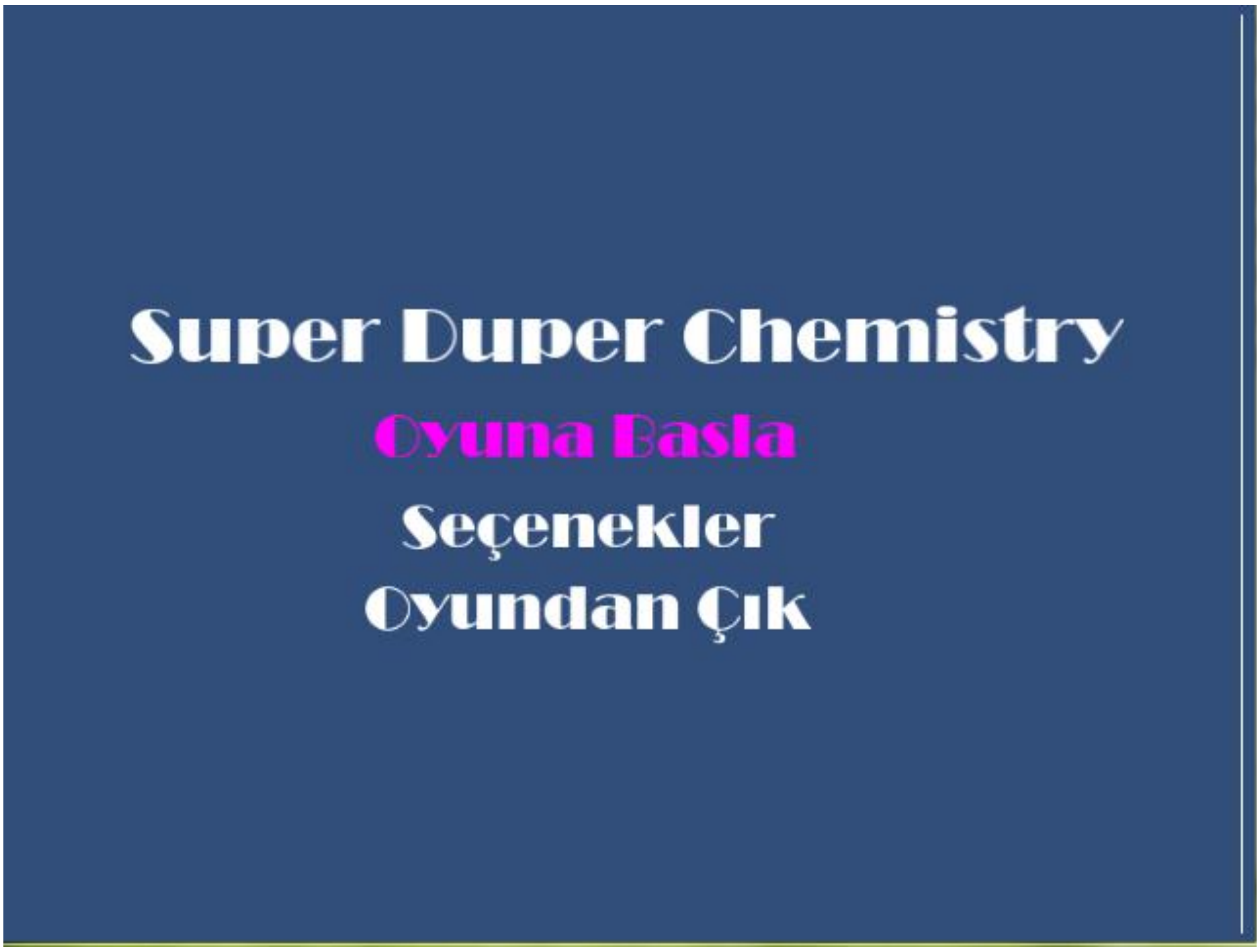


Figure 21: Main Menu screen sample

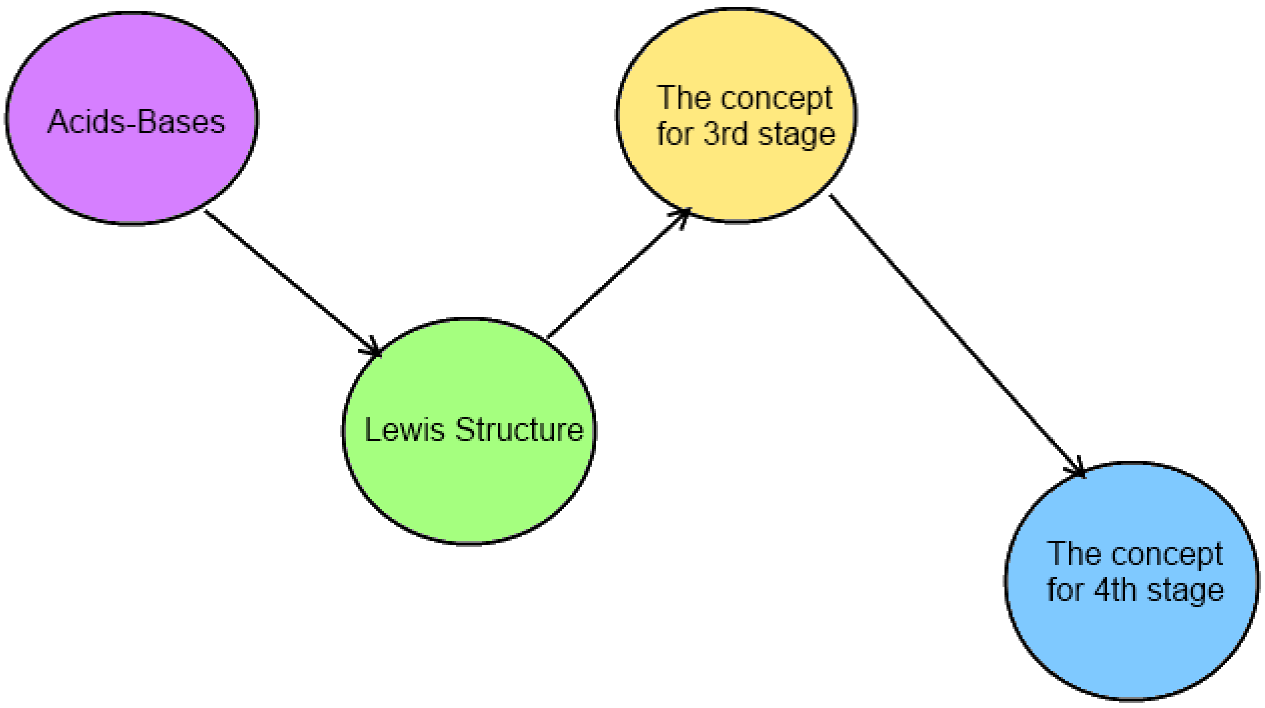


Figure 22: Sample map screenshot

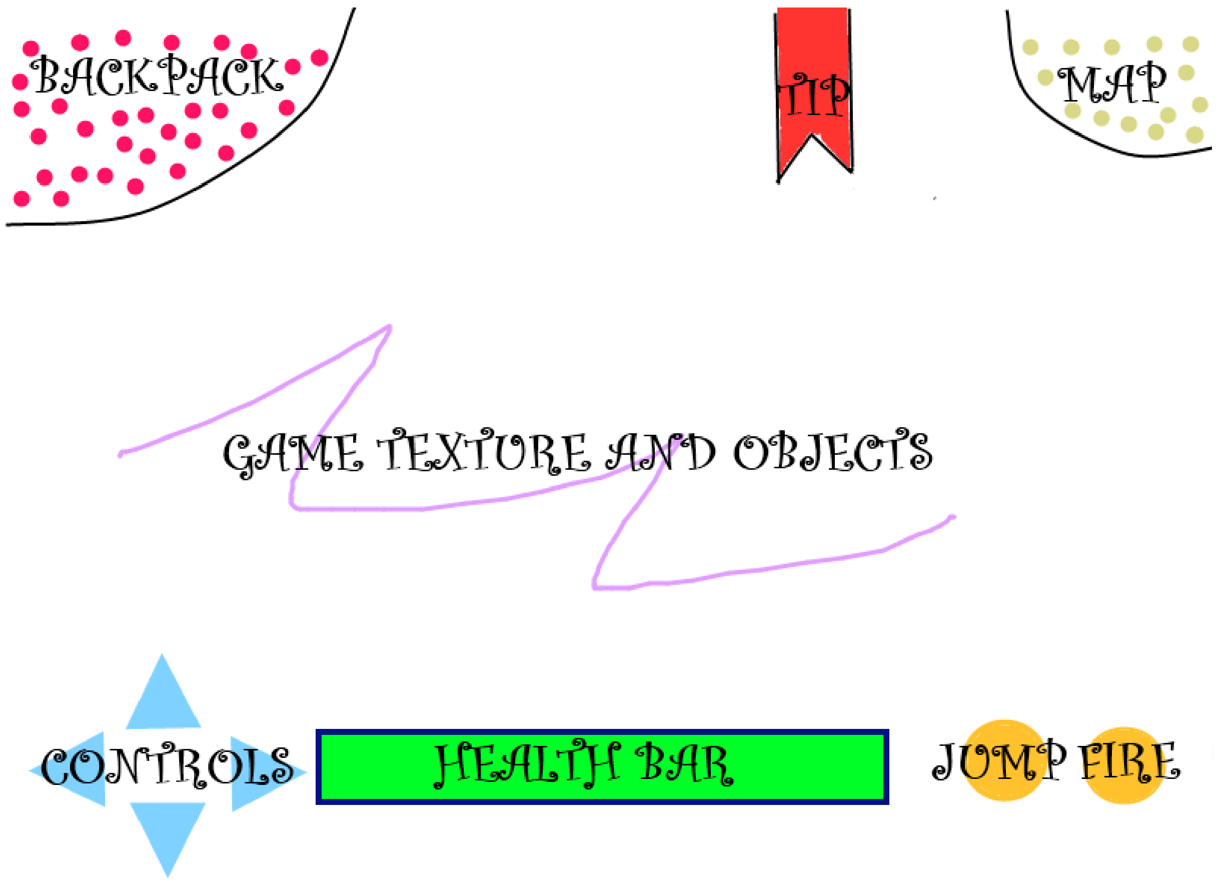


Figure 23: Sample Android game controller screenshot

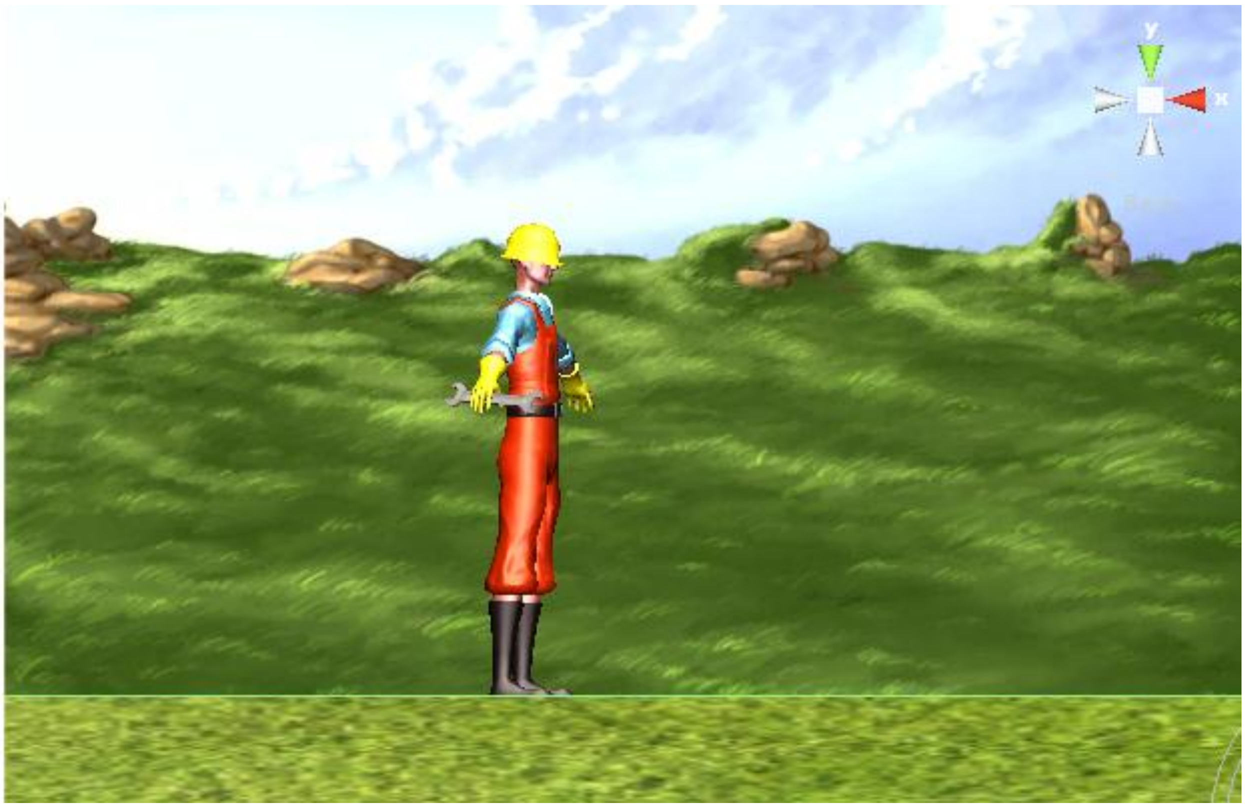


Figure 24: Sample character view screen also can be called as game play screen

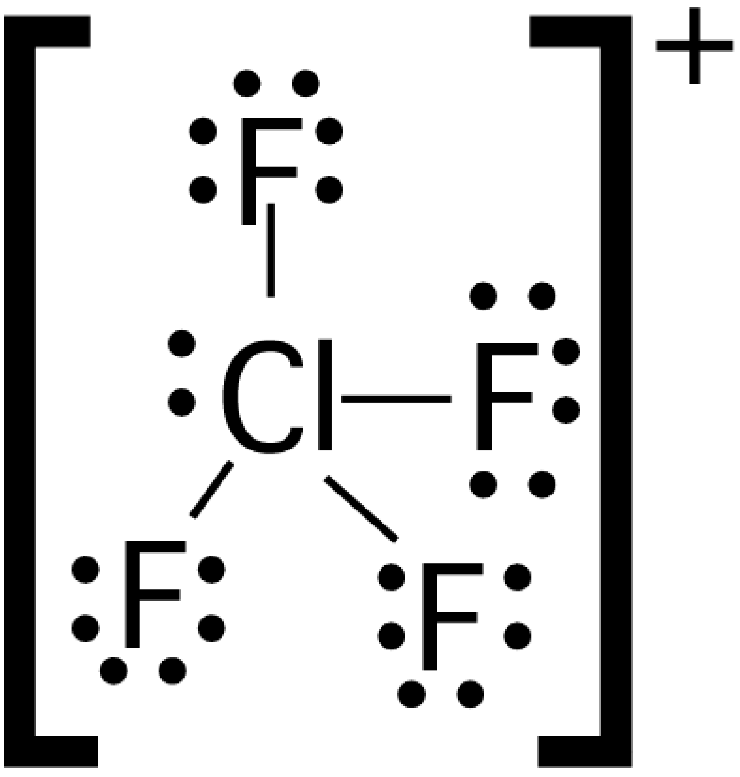


Figure 25: A sample Lewis formula which can be displayed as a puzzle on door

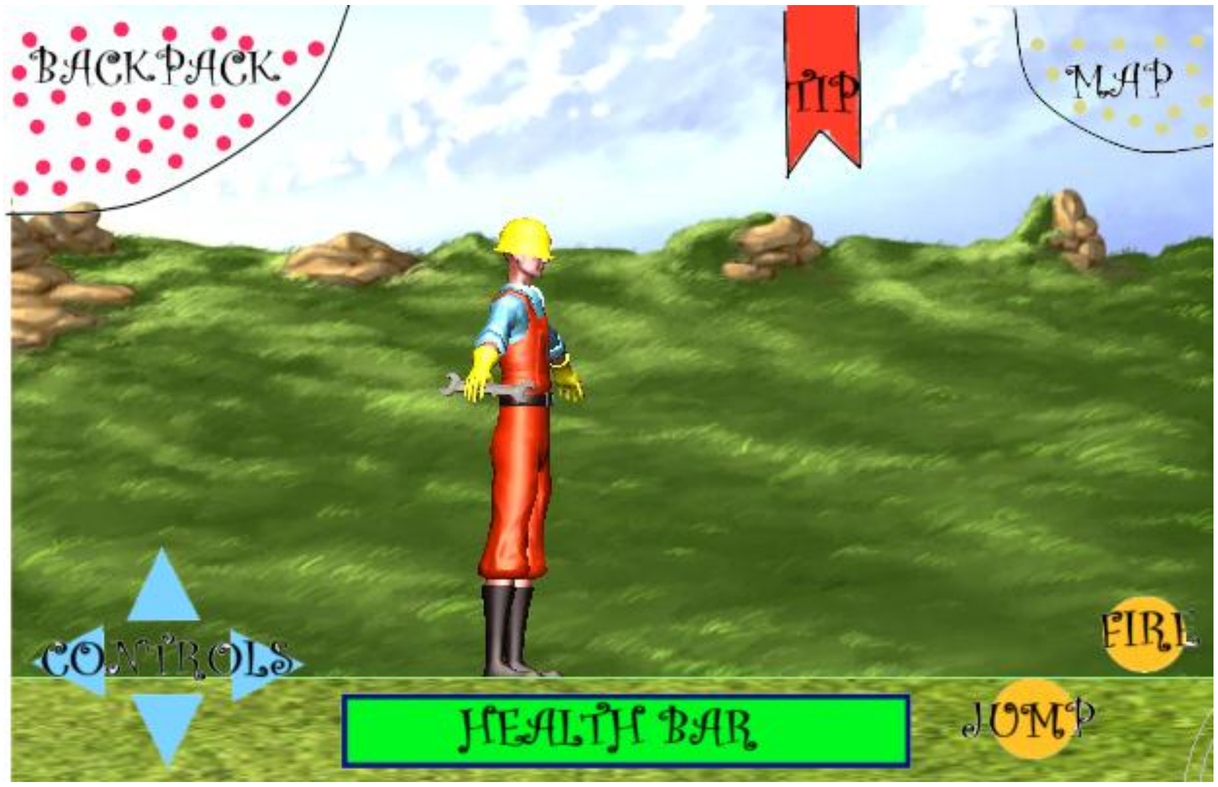


Figure 26: Combined form of figure 23 and figure 24 in Android

6.3. Screen Objects and Actions

In first screen there is going to be :

- Play Game
- Options
- Achievements
- Quit

In the map screen there is going to be :

- Acid and bases
- Lewis Structure
- 3rd concept (going to be decided second term)
- 4th concept (going to be decided second term)

In the game screen there is going to be :

- Direction
- Jump
- Attack
- Map
- Tips
- Backpack
- Appropriate game texture

7. Libraries and Tools

7.1. Libraries

Since we will use the Unity environment, we won’t need any external libraries. But we will still use Unity’s packages which we will explain below.

7.1.1. Character Controller
Character controller package includes the generic controller scripts that can be used for third-person or first-person games.

7.1.2. Light Flares
This package contains necessary scripts to create light flares.

7.1.3. Particles
This package contains necessary scripts to create particle graphics.

7.1.4. Physics Materials
This package has the scripts to adjust friction and bouncing effects of colliding objects.

7.1.5. Projectors
Projectors are used to project materials onto objects. This package helps to use the projectors.

7.1.6. Scripts
The Scripts package contains basic scripts such as camera scripts, general scripts and utility scripts.

7.1.7. Standard Assets (Mobile)
This library includes components specific to mobile environments.

7.1.8. Toon Shading
This library includes the shader scripts to create realistic shading effects.

7.1.9. Water
This package includes the water effect scripts.

7.2. Tools

7.2.1 Unity

Unity is a cross-platform integrated game development tool. Unity differentiates itself with its integrated engine and various deployment platforms. This integrated engine makes it possible to easily test the game where the extensive variety of the deployment platforms enables us to deploy the game to the different platforms without thinking about platform differences.

8. Time Planning

8.1. First Term Gantt Chart

TASK	TEAM MEMBER	October 31	November 7	November 14	November 21	November 28	December 5	December 12	December 19	December 26	January 2	January 9	January 16
Market Research	Emel	🟥	🟥										
Studying Students' Needs	Basak, Emel		🟧	🟧	🟧	🟧							
Technology Research	Cem, Oytun		🟨	🟨	🟨	🟨							
Brainstorming	Cem, Basak, Emel, Oytun		🟩	🟩	🟩								
Design Workshop	Cem, Basak, Emel, Oytun			🟦									
Concept Selection	Oytun, Basak				🟦	🟦	🟦						
Storyboard Design	Cem, Basak, Emel, Oytun				🟪	🟪	🟪	🟪	🟪	🟪			
Structure Design	Cem, Basak, Emel, Oytun				🟪	🟪	🟪						
User Experience Design	Cem					🟪	🟪						
Graphics Design	Volkan Uysal						🟦	🟦	🟦	🟦	🟦	🟦	🟦
User Interface Design	Cem, Emel						🟫	🟫	🟫				
Title - Menu Asserts	Emel							🟧	🟧				
Title - Menu Design	Basak, Cem							🟨	🟨				
Demo Implementation	Cem, Basak, Emel, Oytun								🟩	🟩	🟩	🟩	🟩
Testing Demo	Basak, Oytun											🟥	🟥
Demo Presentation	Cem, Basak, Emel, Oytun												🟦

Figure 27: First term Gantt Chart

8.2. Second Term Gantt Chart

TASK	TEAM MEMBER	February 13	February 20	February 27	March 5	March 12	March 19	March 26	April 2	April 9	April 16	April 23	April 3	May 7	May 14	May 21	May 28	June 4	June 11
Graphics Design	Volkan Uysal	■	■	■	■	■	■	■	■	■	■	■	■	■					
Design of second level	Cem, Basak, Emel, Oytun	■	■	■	■														
Getting Students Feedback about first level	Cem, Basak, Emel, Oytun	■	■	■	■														
Keyboard Mouse Control	Oytun				■	■													
Second Level Implementation	Cem, Basak, Emel, Oytun				■	■	■	■	■	■									
Music Production	Emel, Oytun						■	■	■										
Sound Effects Creation	Emel, Oytun						■	■	■										
Putting Everything Together	Cem, Basak, Emel, Oytun									■	■	■	■	■					
Polishing	Cem, Basak, Emel, Oytun														■	■	■	■	
Getting Students Feedback	Cem, Basak, Emel, Oytun														■	■	■		
Documentation	Emel, Basak, Oytun														■	■	■	■	■
Final Testing	Cem, Basak, Oytun															■	■	■	
Final Presentation	Cem, Basak, Emel, Oytun																		■

Figure 28: Second term Gantt Chart

9. Conclusion

In this initial design report, we improved designs we mentioned in the SRS document. First of all, This report consists of representation of the system, assumptions and dependencies. Then, we defined and explained data structures and architectural components. we have given information about user interface and the libraries. Finally, Gantt chart notation is given at the end of the document. This report is going to be very helpful in the future for understanding and implementing design patterns. To sum up, this initial design report will be the guideline of our project this year.

10. Appendix

Figure Table

Figure: Figure Name	***Page***
Figure 1: System's architectural design	*11*
Figure 2: Components interactions	*12*
Figure 3: Camera Component	*14*
Figure 4: Light Component	*15*
Figure 5: Background Component	*16*
Figure 6: Character Component	*18*
Figure 7: Map Component	*19*
Figure 8: Big Brain Component	*20*
Figure 9: Backpack Component	*21*
Figure 10: Zombie Component	*22*
Figure 11: Element Component	*23*
Figure 12: Camera Component	*24*
Figure 13: Liquid Component	*25*
Figure 14: Litmus Paper Component	*26*
Figure 15: Erlenmeyer Component	*27*
Figure 16: Sun Component	*28*
Figure 17: Salt Component	*29*
Figure 18: Clue Component	*30*
Figure 19: Stone Component	*31*
Figure 20: Door Component	*32*
Figure 21: Main Menu screen sample	*34*
Figure 22: Sample map screenshot	*35*
Figure 23: Sample Android game controller screenshot	*35*
Figure 24: Sample character view screen also can be called as game play screen	*36*
Figure 25: A sample Lewis formula which can be displayed as a puzzle on door	*36*
Figure 26: Combined form of figure 23 and figure 24 in Android	*37*
Figure 27: First term Gantt Chart	*40*
Figure 28: Second term Gantt Chart	*41*