

Chapter Four

Design of Project

4.1 Design

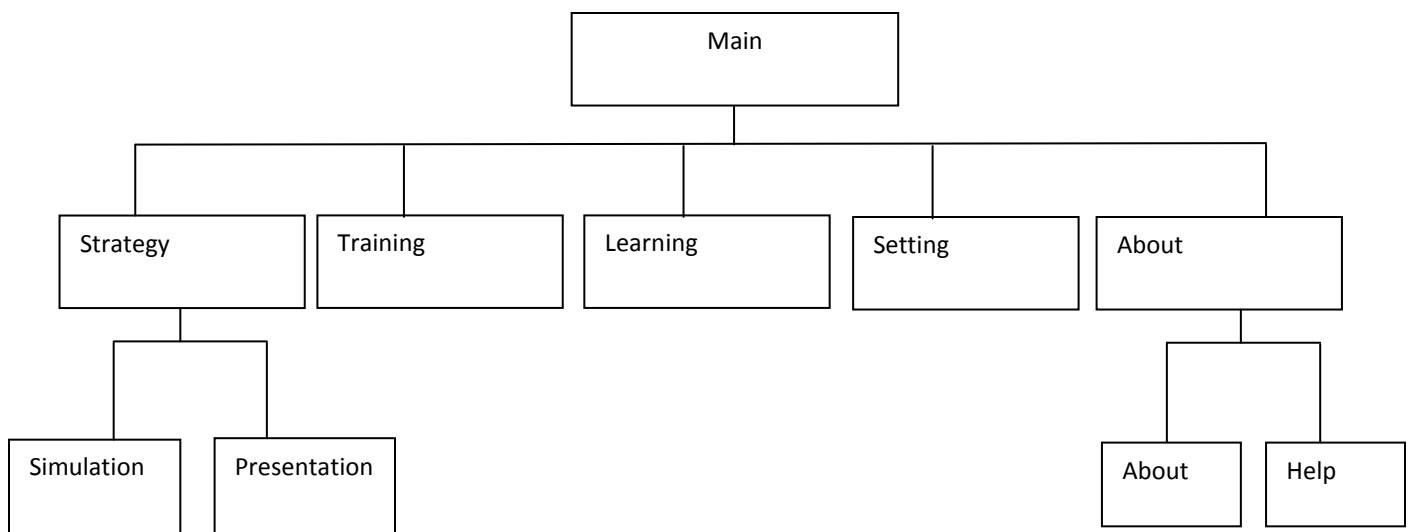
Modeling diagrams help to understand, clarify, and communicate ideas about the code and the user requirements that a software system must support. For example, to describe and communicate user requirements, we can use Unified Modeling Language (UML) use case, activity, class, and sequence diagrams. To describe and communicate the functionality of a system, we use UML component, class, activity, and sequence diagrams.

4.2 Interface Design (Front End)

4.2.1 Design Techniques for Multimedia

Structure Charts is a diagram that describes the way content is organised within an application.

Example of a Structure Chart:



Structure Chart of the main screens in the project.

4.2.2 Storyboard

Storyboards are graphic organizers in the form of illustrations or images displayed in sequence for the purpose of pre-visualizing a motion picture, animation, motion graphic or interactive media sequence.

The storyboarding process, in the form it is known today, was developed at the Walt Disney Studio during the early 1930s, after several years of similar processes being in use at Walt Disney and other animation studios.

1-Main:

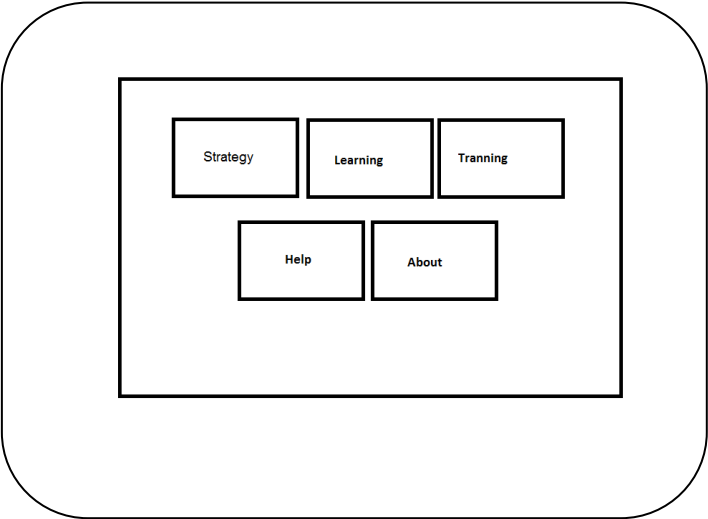
<p>Project:</p> <p>Author: Mazin</p> <p>Screen Name: Main Screen</p>	<p>Interactivity:</p> <p>None</p>
<p>Links from: Main Screen Action: Click strategy button for strategy screen, Click setting for setting screen, Click about for about screen , Click Tanning for Tanning Screen ,Click Learning for Learning Screen.</p>	<p>Video/animation:</p> <p>Camera move in 3d animation</p>
	<p>Audio:</p> <p>None</p>
	<p>Colors:</p> <p>Text:</p> <p>The title for the other screen</p>

Figure 4.1: Main Storyboard

2-Strategy:

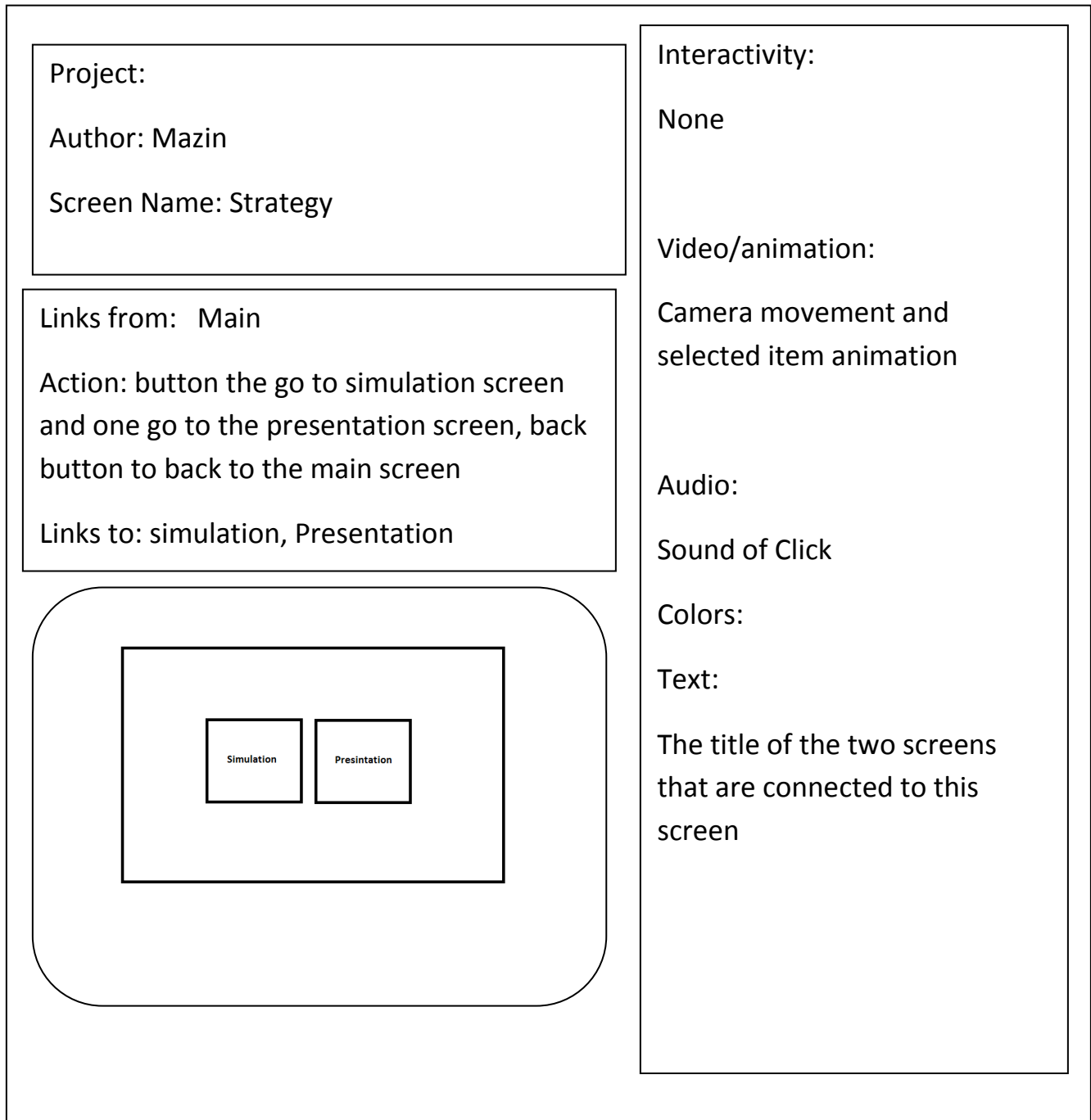


Figure 4.2: Strategy Storyboard

3-Presentation:

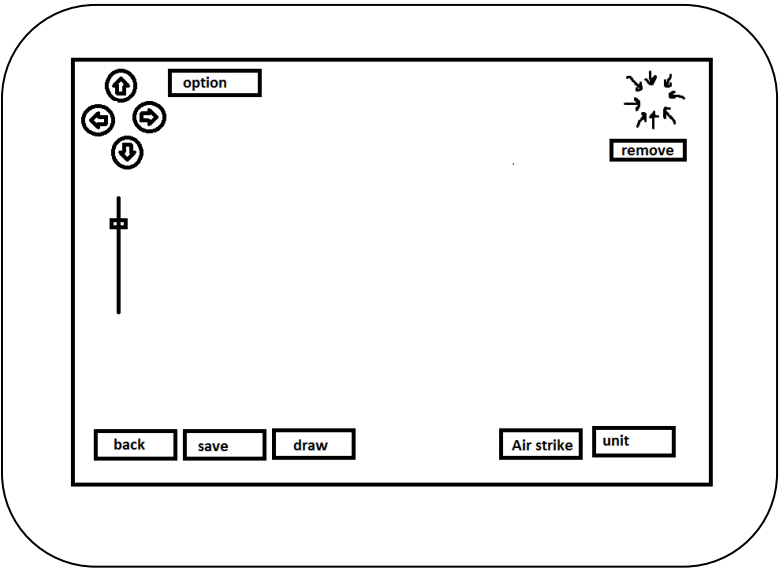
<p>Project :</p> <p>Author: Mazin</p> <p>Screen Name: Presentation</p>	<p>Interactivity:</p> <p>None</p>
<p>Links from: strategy</p> <p>Action:</p> <p>Links to :</p>	<p>Video/animation:</p> <p>Light, plan animation, map movement</p>
	<p>Audio:</p> <p>Sound effect</p> <p>Colors:</p> <p>Text:</p> <p>The arms name and position information</p>

Figure 4.3: Presentation Storyboard

4-Simulation:

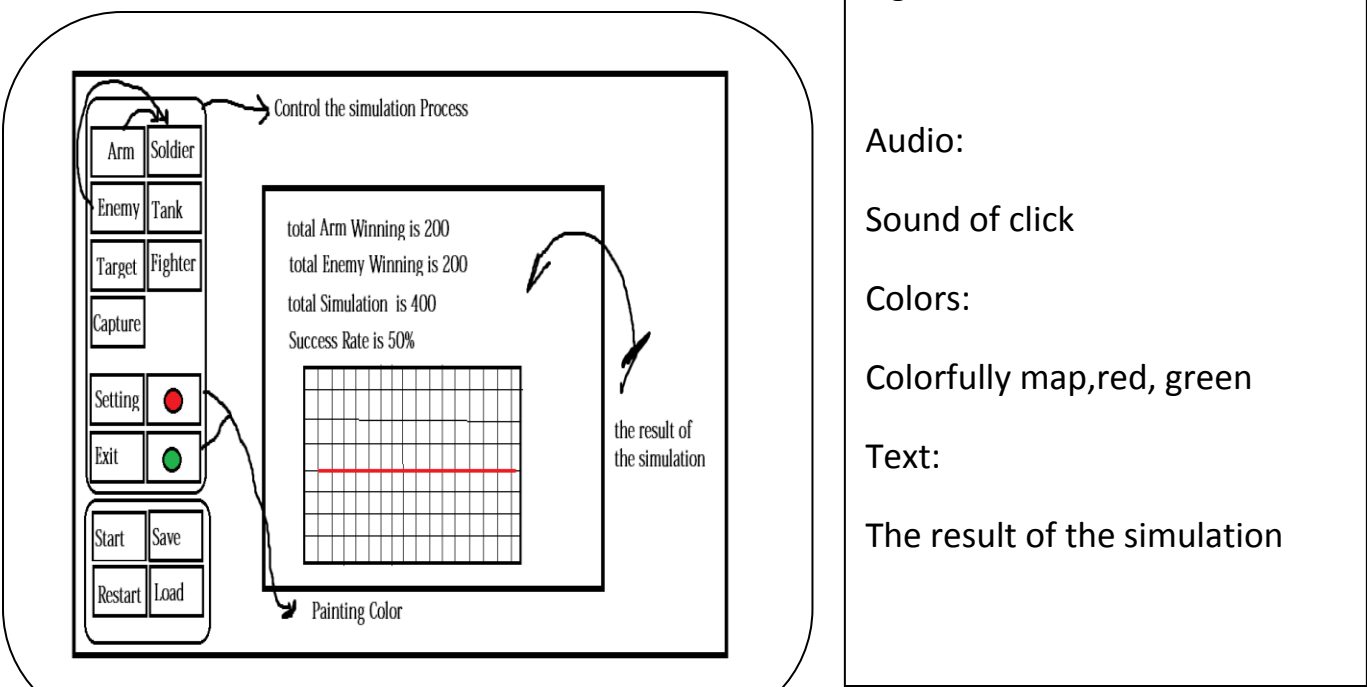
<p>Project:</p> <p>Author: Mazin</p> <p>Screen Name: Simulation</p>	<p>Interactivity:</p> <p>None</p>
<p>Links from: strategy</p> <p>Action: add solder button, add Tank button, add Plan button, add Target button, start/Pause button, restart button, redpaint, green paint, setting button</p> <p>Links to:</p>	<p>Video/animation:</p> <p>Solder movement, Tank movement,</p> <p>Plan movement, camera movement,</p> <p>Light.</p>
	
<p>Audio:</p> <p>Sound of click</p> <p>Colors:</p> <p>Colorfully map, red, green</p> <p>Text:</p> <p>The result of the simulation</p>	

Figure 4.4: Simulation Storyboard

5- Learning:

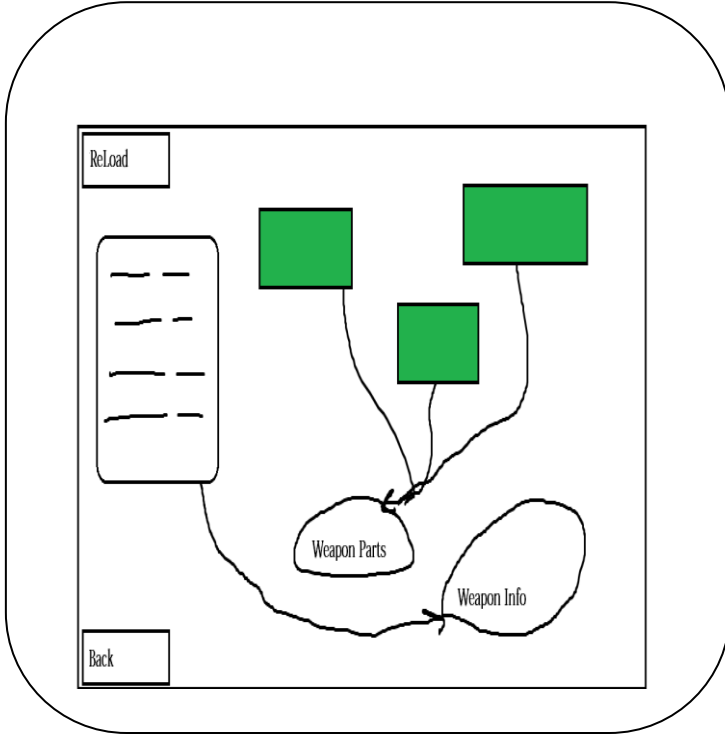
<p>Project:</p> <p>Author: Mazin</p> <p>Screen Name: Learning</p>	<p>Interactivity:</p> <p>None</p> <p>Video/animation:</p> <p>Weapons Parts movement, Camera Movement</p> <p>Audio:</p> <p>Sound of Attach or Connect new Part</p> <p>Colors:</p> <p>Text:</p> <p>Some information about the weapon</p>
<p>Links from: Main</p> <p>Action: Drags and Drop Weapon Parts using mouse, back button</p> <p>Links to:</p>	
	

Figure 4.5: Learning Storyboard

7-About:

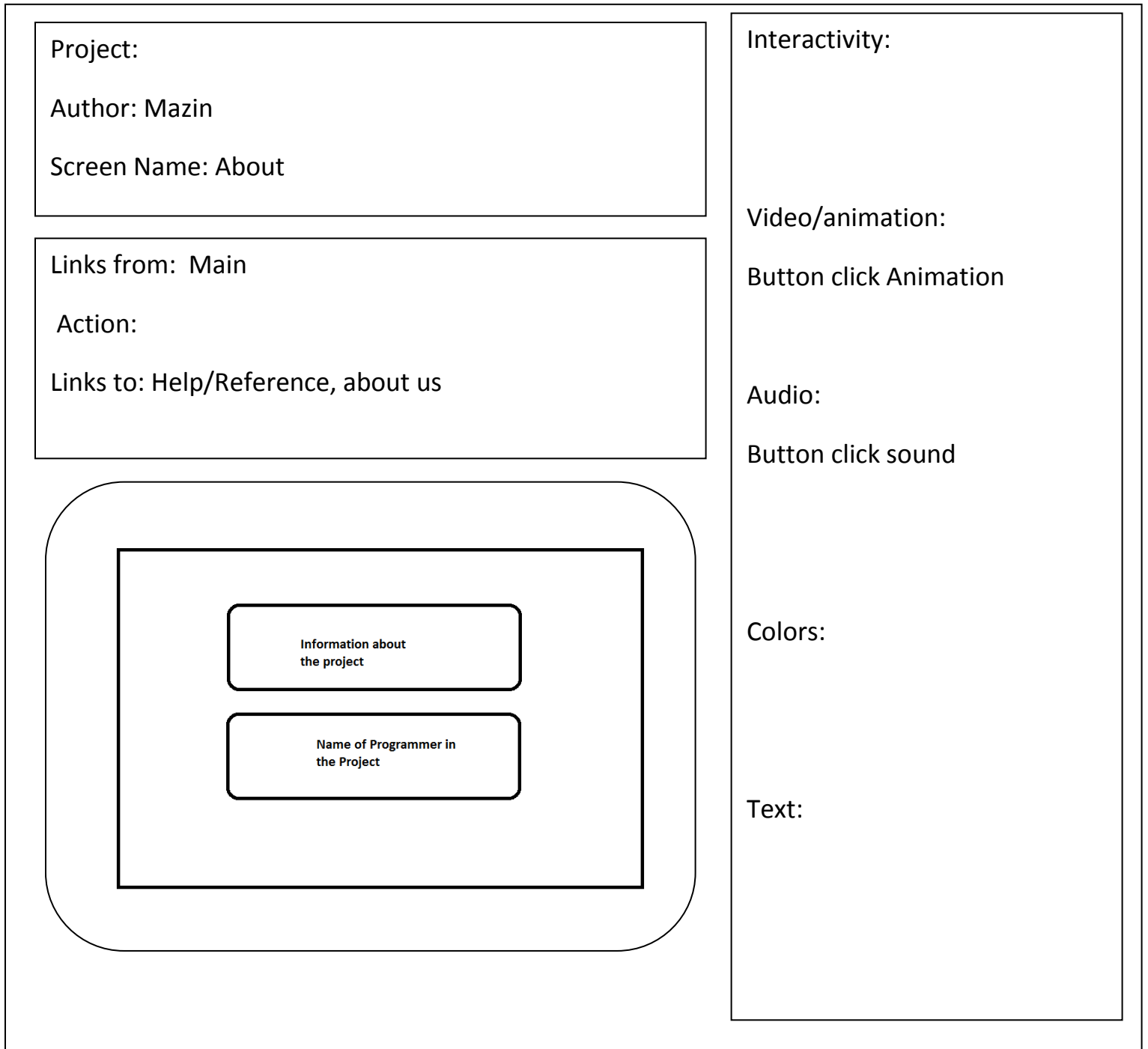


Figure 4.7: About Storyboard

4.3 Back End

4.3.1 Algorithm

A) File Browse algorithm:

1. Declare a variable called Path to store the current path for the directory.
2. Declare a variable called Root to store the previous path in case you want to go back.
3. Declare a variable by name Selected File to handle files.
4. Declare a variable called Extension for purpose of presenting specific files.
5. For each folder in the current path assign to it its graphical icons.
6. For each file with the specific extension assign to it its graphical icons.
7. If the folder is clicked make Root=Path, and Path=Path + folder name.
8. If there's no file selected Return to number 5.
9. End.

B) Analysis algorithm:

It includes four algorithms, and they are:

- 1- Enemy movement algorithm.
- 2- Army movement algorithm.
- 3- Curve algorithm.
- 4- Combat algorithm.

B.1) Enemy movement algorithm:-

1. Declare a variable called target.
2. Declare a variable called army.
3. For each target in map add the target to an array.
4. Sort the array of targets depending on the distance between target and enemy.
5. Assign the variable target to the first target in the array.
6. Check the distance between enemy and army.
7. If the distance of army is closer than the distance of target then attack the army.
8. Else attack the target.
9. End.

B.2) Army movement algorithm:

1. Declare a variable called enemy.
2. Declare a variable called select.
3. If mouse is clicked on an army object make variable Select=true.
4. If mouse is clicked on the map get the mouse position.
5. Convert mouse position from view port space to world space.
6. Tell the army to move to the new world space.
7. Set variable Select to false.
8. If the distance between army and enemy is less than the distance of combat and the level of army weaponry is higher than or equal to enemy weaponry begin attack.
9. End.

B.3) Curve algorithm:

1. Declare a variable called Position Y.
2. For each number in curve array (that is defined in Combat algorithm)
Position Y = Position Y + curve array item[i].
3. Position of x is incremented by one.
4. End.

B.4) Battle algorithm:

1. Declare a static variable called rate.
2. Declare an array called curves.
3. Declare a static variable called total army.
4. Declare a static variable called total winning army.
5. Declare a static variable called total enemy.
6. Declare a static variable called total winning enemy.
7. If an army object is added to the map the variable total is incremented by one.
8. If an enemy object is added to the map the variable total enemy is incremented by one.
9. If army object is destroyed the variable enemy_total_winning is incremented by one and add a value of one to the array of curves.
10. If enemy object is destroyed the variable enemy_total_winning is incremented by one and add a value of one to the array of curves.
11. If simulation is running go to number 6.

12.Else Rate= (total_winning_army/ (enemy_total_winning + total_winning_army))*100.

13.End.

C) Saving algorithm:

1. For each object in the scene get the position and tag of object.
2. Store it in a text file.
3. Encrypt the stored file.
4. End.

D) Kinect algorithm:

1. Declare a variable from type vector 3 called Timer.
2. Declare a variable from type vector 3 called head_Pos.
3. Declare a variable from type vector 3 called left_shoulder_pos
4. Declare a variable from type vector 3 called right_shoulder_pos.
5. Declare a variable from type vector 3 called left_elbow_pos.
6. Declare variable from type vector 3 called right_elbow_pos.
7. Declare a variable from type vector 3 called left_hand_pos.
8. Declare a variable from type vector 3 called right_hand_pos.
9. Declare a variable from type vector 3 called heap_pos.
- 10.Declare a variable from type vector 3 called left_Knee_pos.
- 11.Declare a variable from type vector 3 called right_Knee_pos.
- 12.Declare a variable from type vector 3 called left_foot_pos.
- 13.Declare a variable from type vector 3 called right_foot_pos.
- 14.If Kinect is connected and powered on then start.
- 15.Capture depth image from Kinect.
- 16.Display the depth image.
- 17.For each joined skeleton get its position and assign it to its variable.
- 18.Check if the soldier in correct position then the timer will be incremented by one else timer value=0.
- 19.If timer / 60 >= 5 mints then go to next level.
- 20.Else Return to number 18 and change the position to the next level.
- 21.End.

4.4 Class diagram

Class Diagram provides an overview of the target system by describing the objects and classes inside the system and the relationships between them. It provides a wide variety of usages; from modeling the domain-specific data structure to detailed design of the target system. With the share model facilities, you can reuse your class model in the interaction diagram for modeling the detailed design of the dynamic behavior. The Form Diagram allows you to generate diagram automatically with user-defined scope.

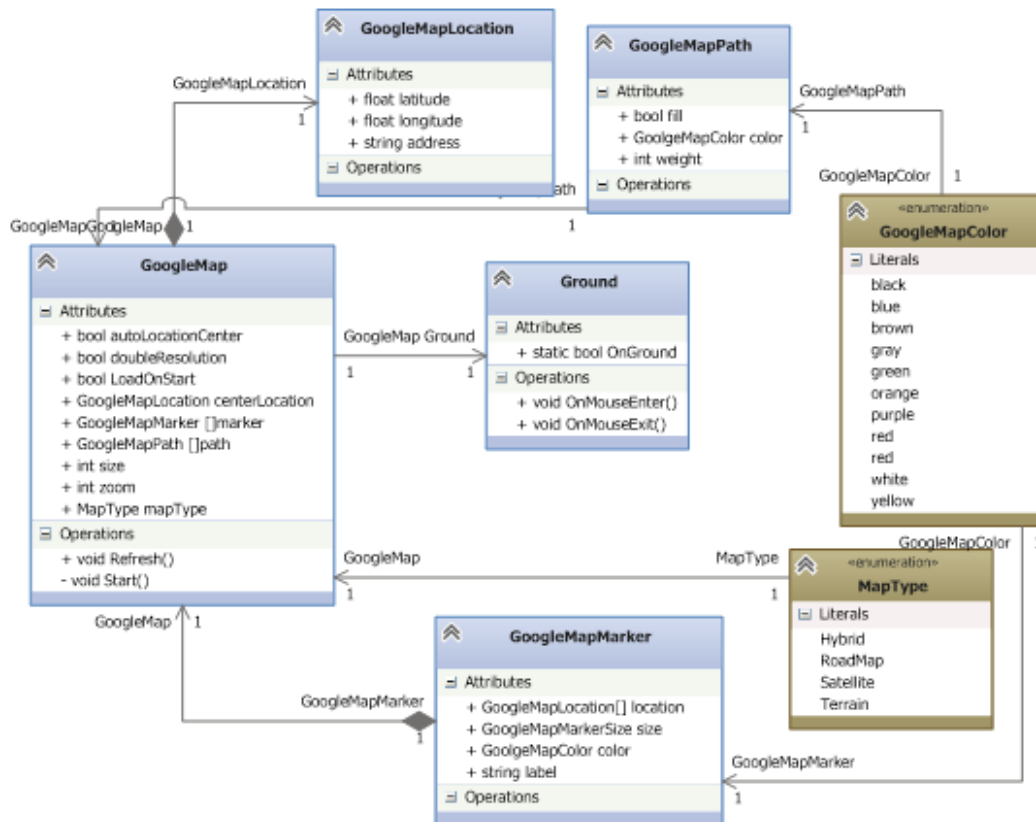


Figure 4.8: Google Map

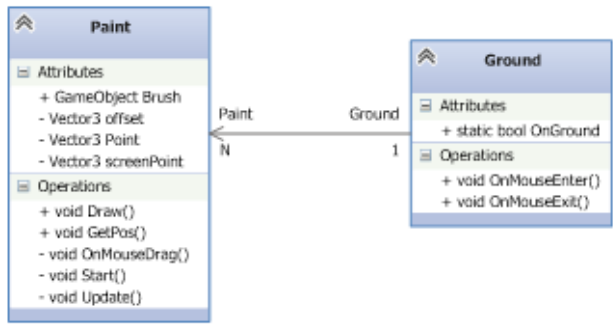


Figure 4.9: Paints

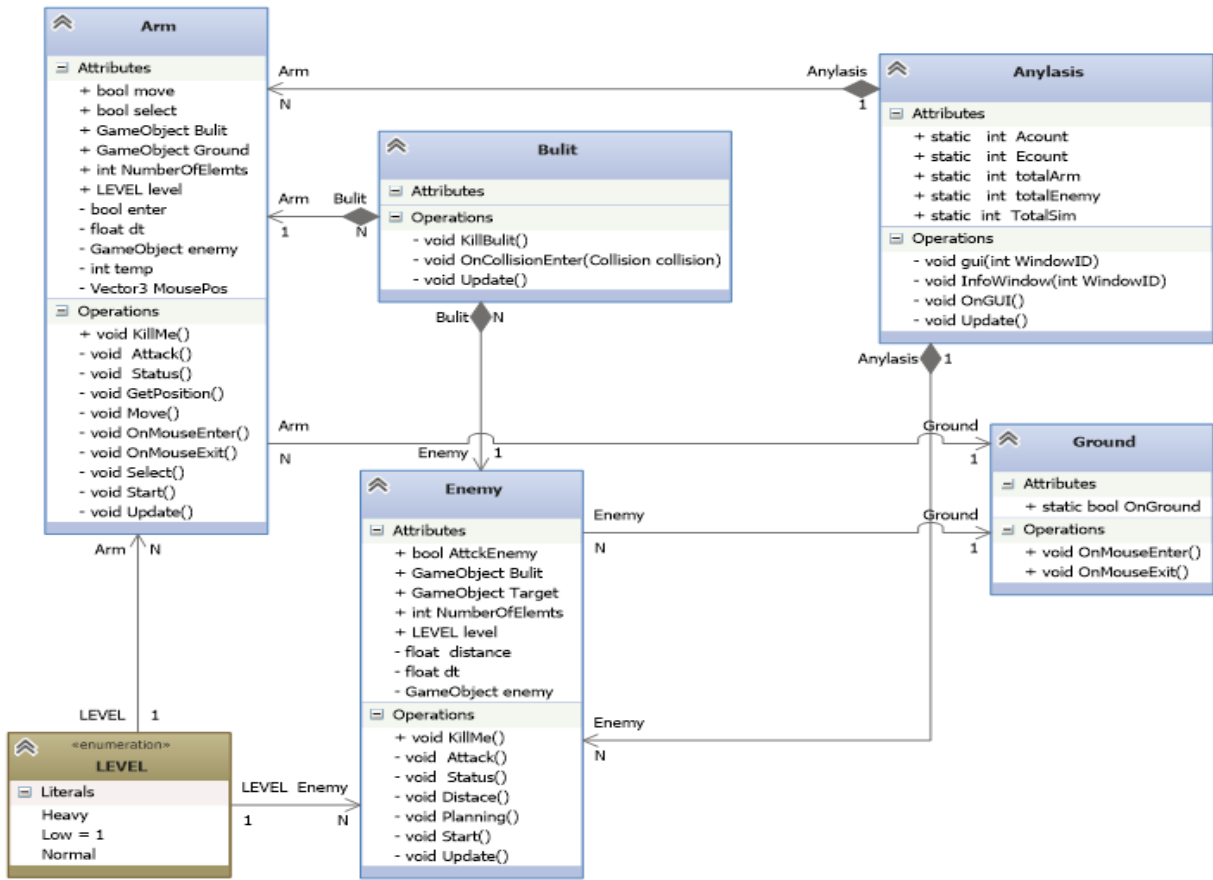


Figure 4.10: Class object

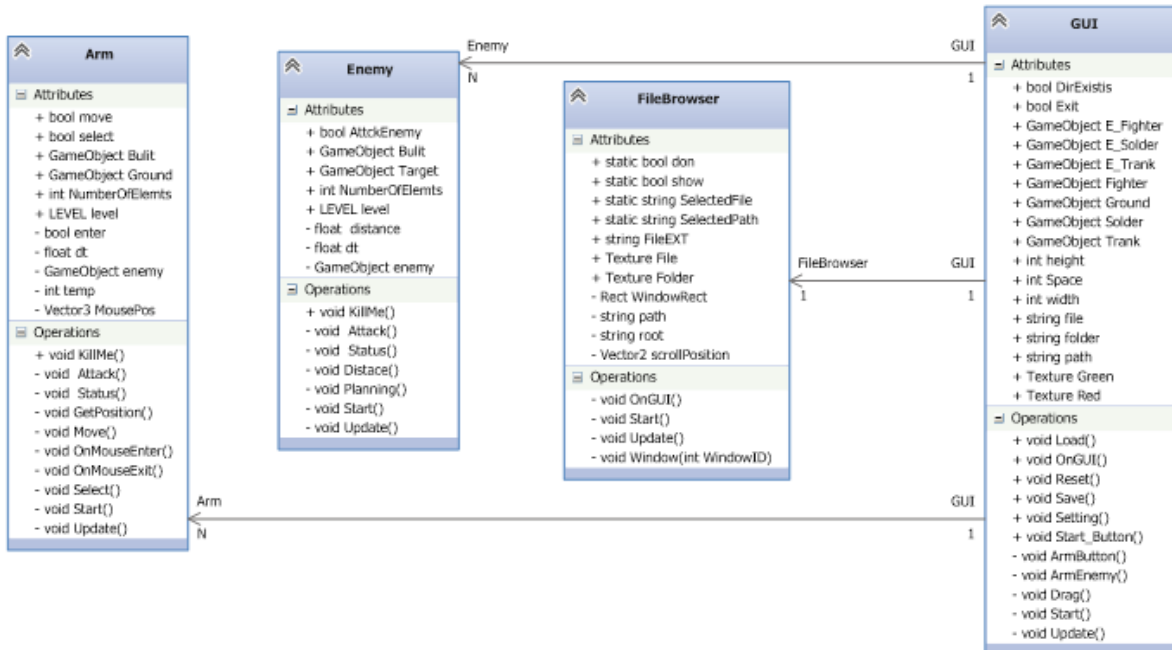


Figure 4.11: GUI

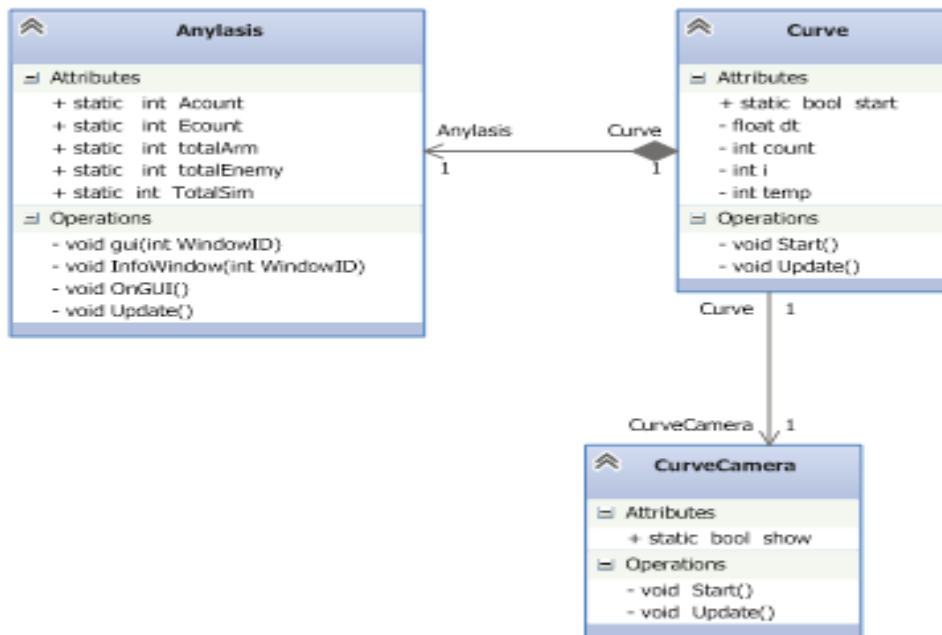


Figure 4.12: Curve