Attack of Township

Submitted by

Md Moniruzzaman (141-35-599)

Supervised By:

Dr. Shaikh Muhammad Allayear

Associate Professor and Head

Department of Multimedia & Creative Technology

Department of SWE

Daffodil International University

Table of Contents:

Chapter: 1

INTRODUCTION

1.1 Project Overview ……1

Chapter: 2

The Purpose of the Project

2.1 Background of the Project……..……..1

2.2 Scope of Our Game……..……...……...1

Chapter: 3

GENERAL DESCRIPTION

3.1 Business perspective ……..1

3.2 System Environment ……..2

Chapter: 4

REQUIREMENTS SPECIFICATIONS

4.1 User Interface ……..……..……..……2

4.2 Hardware Interface ……..…..…..2

4.3 Software Interface…….…..……2

Chapter: 5

Analysis Model

5.1 Use Case Scenario ……..……..6

5.2 Use Case diagram with description ……..…..…….5

5.3 Data Model ……..……..5

5.4 State Diagram……..……..……..6

5.5 Sequence Diagram……...……..6

5.6 Bugs…….....……..……..6

5.7 Patches…….....…….....……..7

Chapter: 6

DESIGN AND IMPLEMENTATION

6.1 Product design terms…….....……...……..……7

6.2 User Experience(UX) …….....……..……..……..7

6.3 Backend Programming ……..……..……...……7

6.4 Level Design…….....…….....……..……8

6.5 System Features…….....……..……..…9

6.6 Construction of the game……...……..9
6.7 Integration with Android……..…….…….…..……9
6.8 Key resource requirement…….....……10
6.9 Implementation tools required……..… . ……10
6.10 Implementation Code Example……...……...……..……11

Chapter: 7
TESTING
7.1 Test Case 1…….……...……14
7.2 Test Case 2…….....…….……14
7.3 Test Case 3…....…….…....……..…..15
7.4 Test Case 4……...…….....…….……..…..15

Chapter: 8
User Manual
8.1 Playing Procedure…….…....……16
8.2 Screenshots…….….…….…….....16

Chapter: 9
Conclusion
9.1 Obstacles …….....……..……19
9.2 Achievements……..…….…….…….19
9.3 Future Plan……..…….....……19

Project overview

In Attack of township players have to collect resource like food (from animals), wood (from tree). Using the resources they can buy/update buildings, troops etc then using those troops they can occupy other players base.

After occupied the base attacker will get certain amount of resource from the defenders until defenders liberate themselves or attacker release them.

Background of the project

Background is a set of events for a plot presented as preceding and leading up to that plot. In our project it's a multiplayer game emphasizing logical thinking and planning. Tactical organization and execution are necessary. Player has to make decision about the actions. All the actions are on the hand of the player.

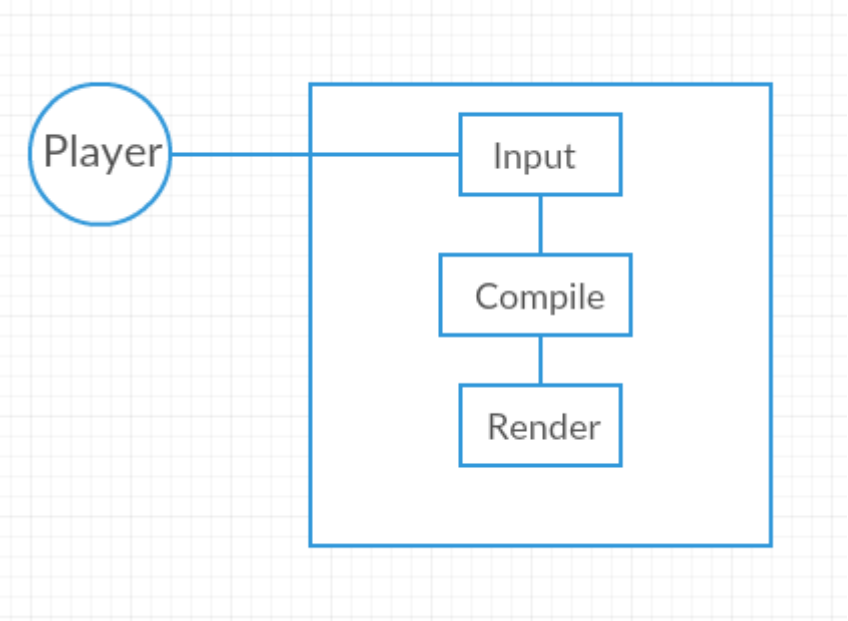
Scope of our game

Project is a multiplayer strategy game on android platform. The purpose of this project is to provide a virtual image of running an empire. Player will learn by the manipulation of the environment through the game. We demonstrate the action flow between input, script and the display. We are working mainly on the levels, animations, objects, scripts. We are not working with the web or IOS platform.

Business perspective

Developing a game from scratch is a much challenging for a company. It requires much investment in terms of effort and cost. We make this game for business purpose as we will distribute this to the client market like google playstore, amazon store etc.

System Environment



Player can interact using the keypad/touchpad as an input then system gives those input to script. Script the will compiled and rendered the result in display.

User Interface

Each game must have menu so that it can be user friendly for the gamer. We have added the menu snapshots in the user manual so that user can understand the menu of the game.

Hardware Interface

“Attack of township” is designed for the android platform. It can be played by android mobiles or tablets. Currently application data is stored locally.

Software Interface

"Attack of township" has been developed using many development tools like-

- Unity 3D
- Blender
- Android Software development kit(Android SDK)

Use Case Scenario

Level 0	Level 1	Level 2
Game	Menu	Buildings
		Shop
		Research
	Train	Train Troops
	Troops	List of Troops
	Attack	Select Friend For Attack

Use Case diagram with description

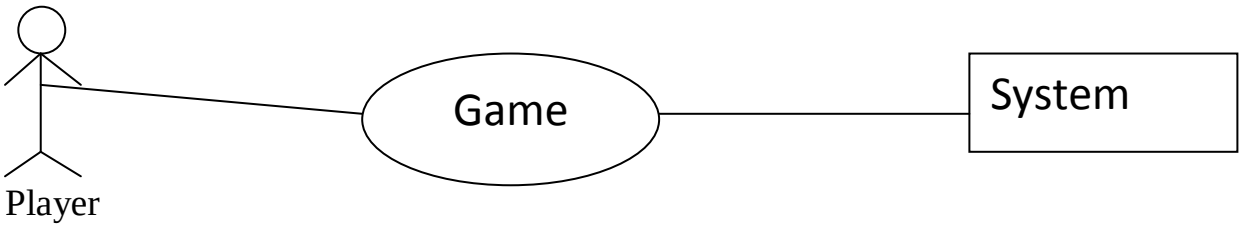


Fig: Level 0 for game

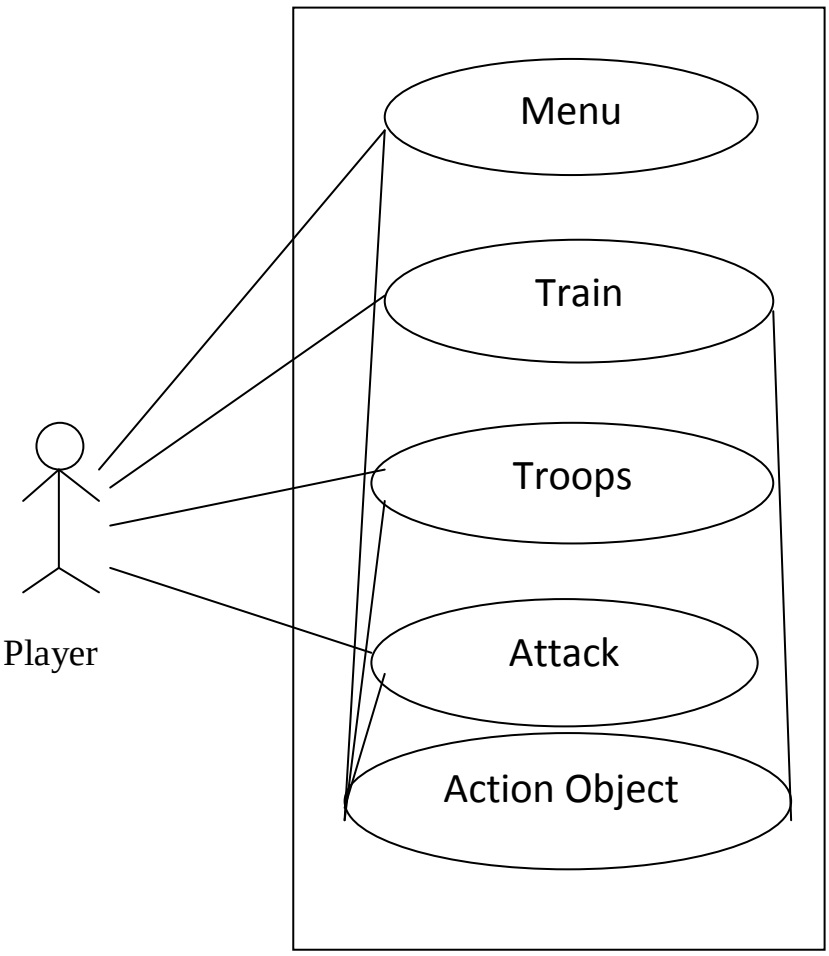


Fig: Level 1 for Game UX

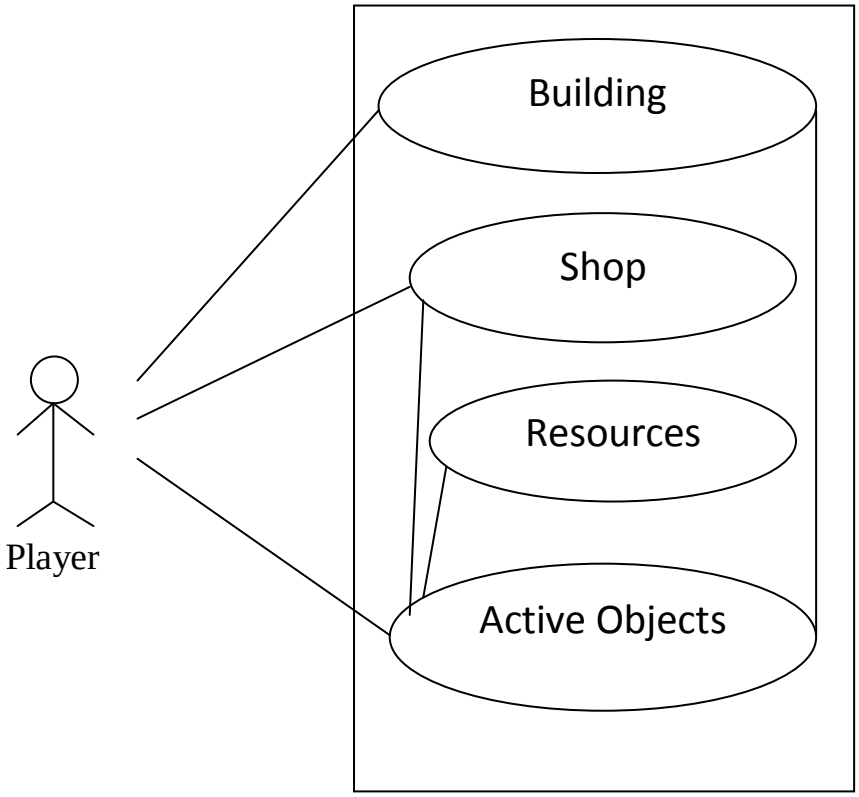
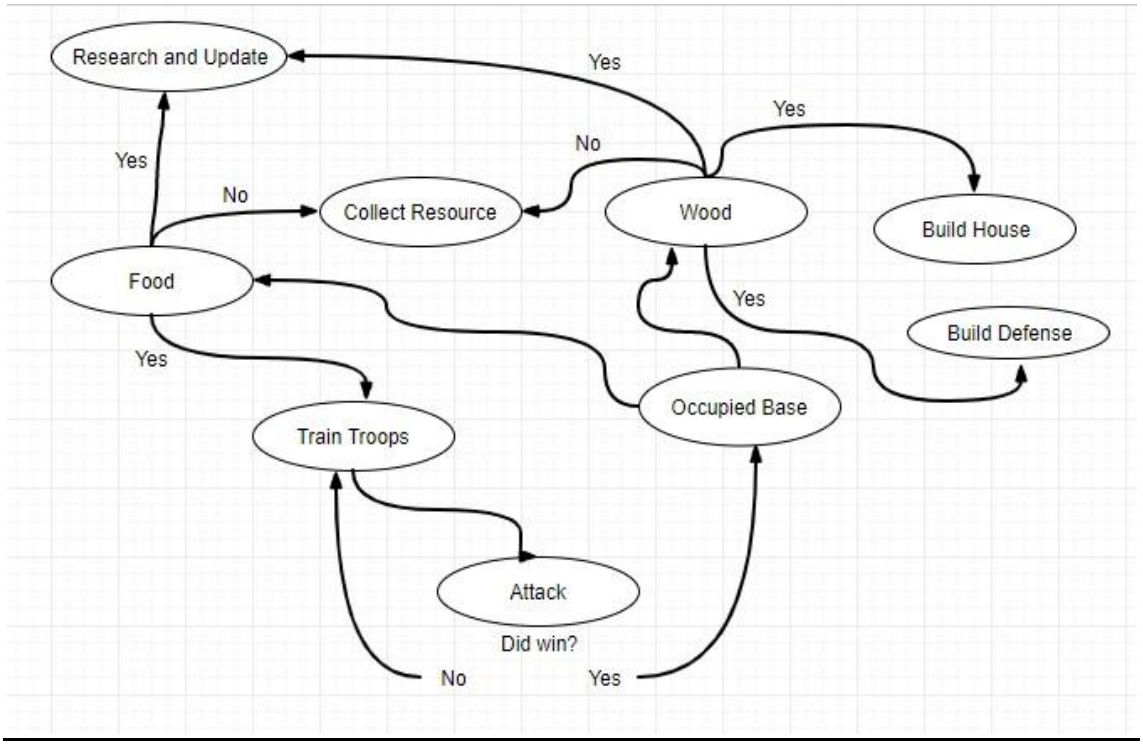


Fig: Level 2.1(Menu) for Game UX

Data Model

If software requirement include the need to create, extend or interact with database then the software team may create a database model for that but although our system has data objects, it does not have any data storage. All the data objects are handled by the game engine so we don't think about data storage that why data model is no need for this application.

State Diagram



Sequence Diagram

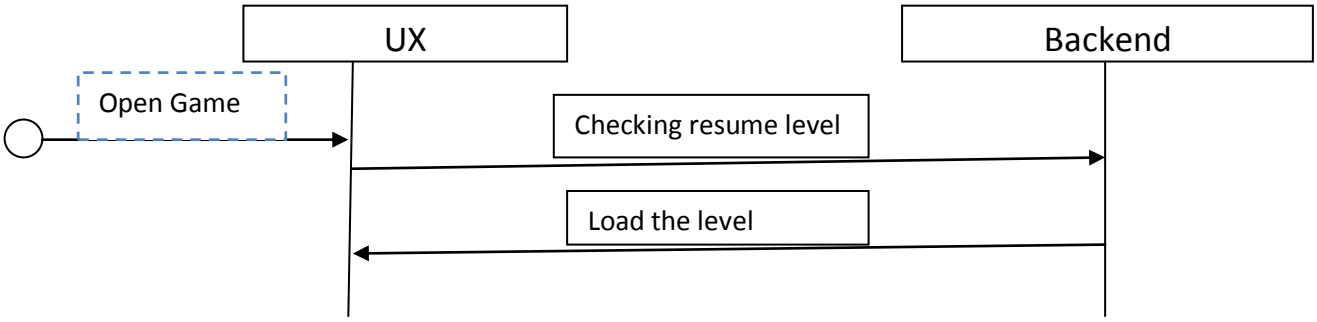


Fig: Resume Game

Bugs

The player will be able to contact the developer by the email system of the application. They can complain if they found any bugs or found anything irrational.

Patches

As the technology is developing day by day the dependencies (like android sdk and also phone configuration) of this project will be updated also. Developers will make changes in order to keep up with any compatibility issue. These changes will be fixed and released through the patches.

Product design terms

For every software product there are two main key things-

- User Experience(UX)
- Backend Programming

User Experience (UX)

Every system must have a frontend so that user can interact with the system. Easy or simple UX design is must for all of the company business and marketing.

UX design is any aspect of user's experience with the given system, including interface, graphics, physical interactions and the manual in most of the cases.

In our game UX is very simple like we have all the menus in front of the screen. Just select the menu and enjoy the rest.

Backend Programming

The backend is the support for the frontend. Backend is responsible for any business logic, accounting logic, database access etc. In simply what users see on screen like player is attacking, the menus, buildings etc is the frontend and what users do not see is backend.
We used c# programming language for programming.

Level Design

Levels	Available Troops and Defense				
I	Police H-40 A-10 AR-5	Rifleman H-60 A-8 AR-30	Worker H-30	Outpost H-100 A-10 AR-40	Wall H-500
II	Sniper H-40 A-40 AR-50	Mortar Infantry H-80 A-10 AR-50 IR-10			
III	Anti Tank H-100 A-1200(t) -10(g) AR-5	Tank H-2500 A-40 AR-30 IR-10			
IV	Aircraft H-3500 A-1200 AR-10 IR-15	Shredder H-1200 A-3000(b, v) AR-0			

H	Health
A	Attack
AR	Attack range
IR	Impact Range
t	Tank
g	General
b	Building
v	Vehicle

System Features

- Collect Resources
- Build Houses
- Upgrade
- Attack Friends
- Generate resources after winning a match
- Timer

Construction of the game

Unity includes many build in components which will make faster of the process of the development. Some of this are-

- Collision detect
- Inputs system
- Object create
- Modify any object
- Scene system
- Physics system
- External model import

Integration with Android

Unity's build in android system makes us much easier to build for android platform. We can create and run for the android platform any time we want. Just need to select from the list which platform we want to build the project then build the project and run it for testing.

Key resource requirement

Project Activity	Skill Required	External Resource
Level Design	Ability to translate story into playable level	Ideas from existing games like clash of clan, age of empire etc.
Physics Engine	Knowledge of unity functions	Unity 3D game engine
Graphics Design	Knowledge of blender functionality and how to use it	Blender
Music	Ability to understand which sound clip is perfect on which situation	From internet resources like soundcloud or youtube
Documentation	Knowledge of report writing	Ideas from existing game reports

Implementation tools required

Product of	Tools	Usage	Use For
Unity Technology	Unity 3D Game Engine	Game Engine	Level or Scene design and all the logics
Blender Foundation	Blender	Model Design	3D models like buildings, trees, characters etc.
Adobe	Photoshop	Picture Editing	Texture edit
Microsoft Windows	Movie Maker	Creating videos	Game cut scenes

Implementation Code Example

Touch Manager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Linq;
public class TouchManager : MonoBehaviour
{
	public static List<Interactive> Selections = new List<Interactive> ();
	public static List<Interactive> AllPlayer = new List<Interactive> ();
	public static Interactive TouchInterectionPlayer;
	// Update is called once per frame
	void Update ()
	{
		if (Input.touchCount == 0)return;
		if (Input.GetTouch (0).phase == TouchPhase.Moved || Input.GetTouch (0).phase !=
TouchPhase.Began) return;
		var es = UnityEngine.EventSystems.EventSystem.current;
		if (es != null && es.IsPointerOverGameObject ()) return;
		var ray = Camera.main.ScreenPointToRay (Input.GetTouch (0).position);
		RaycastHit hit;
		if (!Physics.Raycast (ray, out hit))  return;
		// here check if selections.count>0
		var interact = hit.transform.GetComponent<Interactive> ();
		TouchInterectionPlayer = interact;
		if (interact == null) {
			if(Input.GetTouch(0).tapCount>0){
			Touches.current.DestinationToTouchPosition ();
			}return; }
		if (Selections.Contains (interact)) {return;}
		if (Selections.Count > 0) {
			foreach (var sel in Selections) {
				if (sel != null)
					sel.DeSelect ();
			}
			Selections.Clear ();
			if(interact.GetComponent<Player> ().Info.IsAI){
				Selections.Add (interact);
				interact.Select ();
				return;}}
		foreach (var p in AllPlayer) {
			var distance = Vector3.Distance (p.transform.position,
interact.transform.position);
			if (distance < 30 && p.gameObject.GetComponent<ShowUnitInfo>
().PlayerType == interact.GetComponent<ShowUnitInfo> ().PlayerType) {
				Selections.Add (p);
				p.Select (); }}}}
```

MainMenu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
public class MainMenu : MonoBehaviour {
	public GameObject[] cam; //0=player Camera && 1=Mainmenu Camera
	public GameObject menuContainer;
	public static MainMenu current;
	public Transform CameraDisireLookAt;
	public Transform CameraTransform;
	public Transform BuildingsParent;
	public GameObject BuildingGround;
	//For BuidingPlacement
	public GameObject moveButtonPrefab;
	public GameObject groundPrefab;
	private bool isLoadedFriendList=false;
	// Use this for initialization
	void Start () {
		DontDestroyOnLoad(transform.gameObject);
		activePlayerCamera ();
		current = this;}
	public void activeMainCamera(){
		CameraTransform = cam [1].transform;
		cam [0].gameObject.SetActive (false);
		cam [1].gameObject.SetActive (true);
		menuContainer.SetActive (true);}
	public void activePlayerCamera(){
		cam [0].gameObject.SetActive (true);
		cam [1].gameObject.SetActive (false);
		menuContainer.SetActive (false);}
	public void LookAtMenu(Transform menuTransform){
		CameraDisireLookAt = menuTransform;}
	void Update(){
		if (CameraDisireLookAt != null) {
			CameraTransform.rotation = Quaternion.Slerp
(CameraTransform.rotation,CameraDisireLookAt.rotation,3*Time.deltaTime);}}
	public void BackToMainmenu(int index){
		SceneManager.LoadScene (index);
	}

	public void BuildingPlacement(GameObject buildingPrefab){
		GameObject go=Instantiate (buildingPrefab,
					MainMenu.current.BuildingsParent.position,
					MainMenu.current.BuildingsParent.rotation,
					MainMenu.current.BuildingsParent) as GameObject;
```

```
                GameObject ground = Instantiate (MainMenu.current.groundPrefab,go.transform);
                MainMenu.current.BuildingGround =ground;
                go.GetComponent<Buildings> ().buildingGround = ground;
                MainMenu.current.activePlayerCamera ();
                MainMenu.current.AddToList (go.GetComponent<PlayerHealth>());
}

        void AddToList(PlayerHealth building){
                GameManager.current.attackerBuildingsList.Add (building);

        }

        public void ClosePanel(GameObject panel){
                panel.SetActive (false);
        }

        public void ActivePanel(GameObject panel){
                panel.SetActive (true);
        }

        public void ShowPanel(GameObject panel){
                panel.GetComponent<RectTransform> ().offsetMin = new Vector2 (0,0);
                panel.GetComponent<RectTransform> ().offsetMax = new Vector2 (0,0);
        }

        public void HidePanel(GameObject panel){
                panel.GetComponent<RectTransform> ().offsetMin = new Vector2 (1000,1000);
                panel.GetComponent<RectTransform> ().offsetMax = new Vector2 (1000,1000);
        }
        public void LoadFriendList(GameObject friendsPrefab){
                if (!isLoadedFriendList) {
                        GameObject go = Instantiate (friendsPrefab,
                                        AttackPanel.current.friendListsParent.position,
                                        AttackPanel.current.friendListsParent.rotation,
                                        AttackPanel.current.friendListsParent) as GameObject;
                        isLoadedFriendList = true;
                        go.GetComponentInChildren<Text> ().text = "Munna";
                        go.GetComponent<Button> ().onClick.AddListener
(LoadBattleGroundScene);}}
        public void LoadBattleGroundScene(){
                BackToMainmenu (1);
        }}
```

Test Case 1

Test Case:	This test will check if resource menu is working properly or not.
Test Procedure	**:** Select the mainmenu prefab and drag it into the scene then run the scene.
Expected Result	**:** Can view the building, shop menu.
Actual Result	**:** Cannot view the menu.
Comment	**:** Need to check mainmenu script is attached with the mainmenu prefab or not.
Conditional Test	**:** Again run the scene.
Expected Result	**:** Can navigate to the resource menu**.**
Actual Result	**:** Yes it is working.
Accuracy	**:** Perfectly Accuracy.

Test Case 2

Test Case	**:** This test will check if building can be selected to move or not.
Test Procedure	**:** Go to the resource menu and on building menu select any available building.
Expected Result	**:** Building will be instantiated and also can be moved around the scene.
Actual Result	**:** Run time exception.

Comment **:** Need to check reference building is attached to the script.

Conditional Test **:** Again run the scene.

Expected Result **:** Can instantiate and move the building**.**

Actual Result **:** Yes it is working.

Accuracy **:** Little bit lacking.

Test Case 3

Test Case **:** This test will check if audio is working or not.

Test Procedure **:** Run the start scene.

Expected Result **:** Can listen the sound.

Actual Result **:** Can listen the sound.

Accuracy **:** Perfectly Accuracy.

Test Case 4

Test Case **:** This test will check if attacking is working properly or not.

Test Procedure **:** Select the player prefab and drag it into the scene then run the scene.

Expected Result **:** Player is attacking to the turrets.

Actual Result **:** Attacking perfectly.

Playing Procedure

Gamers first interact with the UI to start playing. We provide playing tips to all the users so that they can easily understand the game.

Firstly players have to open the game application then they have to train troops like villagers (for collecting food/wood), polices (for attack other players) etc. Then villagers will collect food/wood from the nearest food garden or tree. Using that food and wood they can buy more buildings and upgrade that. When player feel that they have enough troops to attack then they can attack other players. If player win the battle then they will be rewarded by food or wood.

Fig: Game Environment

Fig: Resources Menu

Fig: Train Troops Dialogue

Fig: Troops List

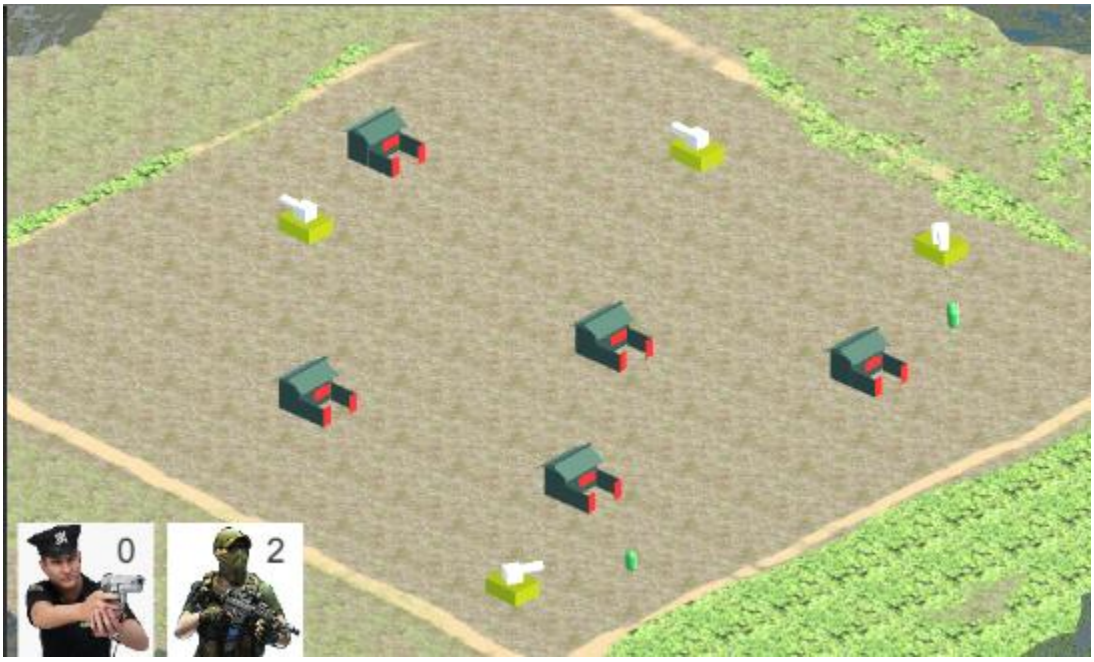


Fig: Attacking opponents

Obstacles

1. Working with the unity is completely new experience to us.

2. We adopt these things by video tutorials. It's a matter of huge time and patience.

3. Creating 3D Model is so difficult.

4. For pathfinding algorithm and attacking we used some packages and that was also difficult for us to find the right package with the right algorithm.

Achievements

1. We know much more about the game engine that how objects work.
2. We know how create 3d model and how to animate it.
3. Growing creative thinking and imagination capability.

Future Plan

- Feature extensions
- Improve UX
- Expands servers capabilities
- Website and social networking pages for promo code of food, wood etc