

**Games Programming**

Year 2013/14, Semester 6

**SCHOOL OF INFOCOMM TECHNOLOGY**

Diploma in Information Technology

**Games Programming**

**ASSIGNMENT1**

Dead Survival

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Student ID** | **Grade** |
| Saklani Pankaj | S10117228A |  |
| Tan Yong Wei | S10117054B |  |

Table of Contents

[**SCHOOL OF INFOCOMM TECHNOLOGY** 1](#_Toc374308904)

[**Implementation of Game Features** 3](#_Toc374308905)

[Student Name: Saklani 3](#_Toc374308906)

[Bullet Appearing at Barrel of Gun 3](#_Toc374308907)

[Random Spawning of Trees 3](#_Toc374308908)

[Mini-map 3](#_Toc374308909)

[GUI Elements Creations 3](#_Toc374308910)

[Sound Control 3](#_Toc374308911)

[3 Types of Zombies 3](#_Toc374308912)

[Zombie Path Finding 3](#_Toc374308913)

[Zombie AI Choosing Target for Attack 3](#_Toc374308914)

[Zombie Homing to Player 3](#_Toc374308915)

[Zombie Attacking 3](#_Toc374308916)

[3 Types of Weapons 3](#_Toc374308917)

[Weapon Switching 3](#_Toc374308918)

[Muzzle flash at the tip of barrel of gun when shooting 3](#_Toc374308919)

[Help Screen 3](#_Toc374308920)

[Building Structures in Game 3](#_Toc374308921)

[Pausing of Game 3](#_Toc374308922)

[Random Spawning of Supply and Ammo Cache 3](#_Toc374308923)

[Shooting recoil 3](#_Toc374308924)

[Student Name: Tan Yong Wei 3](#_Toc374308925)

[Animation for button 3](#_Toc374308926)

[Animation For Player 3](#_Toc374308927)

[8 Direction Movement 3](#_Toc374308928)

[Player Upgrade System 3](#_Toc374308929)

[Leaderboard 3](#_Toc374308930)

[Main Menu 3](#_Toc374308931)

[Audio/Music 3](#_Toc374308932)

[References 3](#_Toc374308933)

# **Implementation of Game Features**

# Student Name: Saklani

### Bullet Appearing at Barrel of Gun

* **Status:** Completed
* **Overview:**  If no calculations are done the bullet when fired from the player appears to come out of the player’s sprite center, which is not the location of the gun barrel. Hence it looks odd. Hence calculations are done to find out the position of the gun barrel in relation to the center of the player sprite.
* **How it was implemented:** Two Operations lengthdir\_x and lengthdir\_y is used to find the x and y origin of the gun barrel in relation to the origin of the player sprite. Both function return the x and y values after a certain length is added to a direction. Hence the length is found by using trigonometry to find the length between the origin and the fun barrel, which is then used in the functions to get the x and y values of the gun barrel in respect to the origin.

### Random Spawning of Trees

* **Status:** Completed
* **Overview:** At the start of the game the Trees in the Map will all spawn at random locations around the map besides the center, which needs to be empty for the player to build structures. All the tree will also have random sprite and will all be at least 120 pixels apart from each other. They will also not spawn outside the room and not spawn on top of each other.
* **How it was implemented:** In order to achieve this feature, in the control object’s room start event, all the instances for the trees are created, at the same exact spot. In the tree object, if the tree object collides with itself, it is move to a random location using the Game Maker function move\_random. In the tree object’s step event, if the tree is outside the room or within the build mode’s area or not 120 pixels away from the nearest tree, it is again moved to a random area using the move\_random function.

### Mini-map

* **Status:** Completed
* **Overview:** The Player will have a mini map at the bottom right hand corner of the room, which would be focused on the house. As the Player needs to explore, he also needs to make sure that the House also survives in the process. Therefore, using the map he will be able to see if the house is being attacked and defend it if necessary.
* **How it was implemented:** In the game room, besides having a main view, which follows the player around, a second view was initialized. The second view has a view height of 1080pixels and view width of 1920pixels. However, the port width was set to 320 pixels and height to 180 pixels, the port x and y position was set to 1600 pixels and 900 pixels respectively. The border for the Mini-map was later added using the control object, as a sprite using the draw GUI event.

### GUI Elements Creations

* **Status:** Completed
* **Overview:** Creating the GUI components for the game using Photoshop.
* **How it was implemented:** Using the Adobe Tool Photoshop all the GUI components such as Title, backgrounds, buttons were all created using the tool. All sprite and visual assets from the internet were also modified to preference using Photoshop.

### Sound Control

* **Status:** Completed
* **Overview:** Allowing the Player to Change the Volume for all the Music and the Sound effects in the Game.
* **How it was implemented:** The Changing of the volume is implemented by using a slider. A background with the slider’s background is created first which is used in a room called option. A object called obj\_Ball is then created, whose position on the slider background, determines the Volume of the sound.

### 3 Types of Zombies

* **Status:** Completed
* **Overview:** The game would have 3 types of zombies with distinct perks and weaknesses, in order to make the game more interesting. The first zombie is the normal zombie that has a normal attack and movement speed. The second is the sprinter, that has more movement speed compared to the normal, but would have lower Health. Lastly the last Zombie is the Tank, which has the highest Health and the highest damage but is the slowest and the biggest of all the zombies
* **How it was implemented:** A main zombie object was made, which has a zombie sprite connected to it. The sprite has the 3 images of all the Zombies, when the zombie is created, the image\_index of the sprite is selected. Depending on the image\_index, the Zombie’s distinct attributes are set.

### Zombie Path Finding

* **Status:** Completed
* **Overview:** The Zombie is able to navigate through the Obstacles on the map and not get stuck. It is also able to unstuck itself, if it intercepts an obstacle, mainly the tree objects.
* **How it was implemented:** The game maker code for path finding mp\_Potential\_Step is used to set the path of the zombie to move towards the house if the Player is not nearer to the zombie.

### Zombie AI Choosing Target for Attack

* **Status:** Completed
* **Overview:** The Zombie makes decision on who to go after depending on the distance of the Zombie to the various objects.
* **How it was implemented:** This is done in the step event of the Zombie object. If the distance of the Zombie to the Player is less than 600 pixels, otherwise the Zombie will go after the building that the player built in the Build Mode. If there are no buildings left than the Zombie will finally go after the House.

### Zombie Homing to Player

* **Status:** Completed
* **Overview:** The Zombie is successfully home to the Player and look at him while doing so.
* **How it was implemented:** In the step event for the Zombie, if the Player is in range, the Zombie sprite’s image\_angle is set to the degree returned from the function point\_to\_object(). After which the function mp\_Potential\_Step is used to make the Zombie move towards the Player without colliding with any obstacles.

### Zombie Attacking

* **Status:** Completed
* **Overview:** The Zombie Attacks at a constant rate, whenever it is in contact with a building the house or the player
* **How it was implemented:** In the Zombie object, a variable called collide is used to check if damage should be given to an object if in contact with the Zombie. Whenever the Zombie collides damage is given if the alarm[0] is lesser or equal to 0, after which, the alarm[0] is set to 60 if the contact is maintained.

### 3 Types of Weapons

* **Status:** Completed
* **Overview:** The Player would have 3 types of weapons and would be able to use them to defend against the waves of zombies. The pistol has infinite ammo, while the other two, the rifle and the shotgun, can only be used when the player has ammo is greater than 0.
* **How it was implemented:** The player would have a gun variable, which stores the value of the gun that the player is currently using. In the Step event of the player, when the player clicks the Left Mouse button, which is used to shoot, depending on the gun variable bullets are created at the barrels of the gun. Therefore, when the pistol is selected a single bullet is created and then the alarm that is used to regulate the fire rate is called, when the rifle is fired, the damage of the bullet is increase, while the alarm’s value is decreased to increase fire rate. As for the shotgun, three bullets are created, with all more damage than the pistol but lower fire rate than the rifle.

### Weapon Switching

* **Status:** Completed
* **Overview:** The player is able to change weapon on command.
* **How it was implemented:** Whenever the mouse scrolled up or down, the gun variable is added by 1, which changes the current weapon to the next, when the last weapon is selected, the gun variable is set back to 0, which is the first weapons. The Numbers 1,2,3 are also used to switch between the weapons.

### Muzzle flash at the tip of barrel of gun when shooting

* **Status:** Completed
* **Overview:** A Muzzle Flash is create at the barrel of the gun, whenever, the gun is shot.
* **How it was implemented:** In the draw method of the player, whenever a the variable flash is true, using the draw\_sprite\_ext() function, the muzzle flash sprite is drawn at the tip of the gun’s barrel.

### Help Screen

* **Status:** Completed
* **Overview:** A Help screen is available for the player to look at what he is supposed to do in the game and the goals and that various items and buildings in the game.
* **How it was implemented:** A room called Help is created which hold a object called obj\_Help, which has a sprite, whose images simulate pages in the room. Which are changed when the left and right arrow are pressed, to change the pages of the help screen.

### Building Structures in Game

* **Status:** Completed
* **Overview:** The Player is able to enter a Build Mode using a button. After which he is able to build building to defend the house against the waves of zombies.
* **How it was implemented:** A third view is initialized in the game room, which focuses around the House. When the Key ‘F’ is pressed, the Game Pauses and the screen focuses on the House. After which the state of the game is changed to Build Mode. An object Called obj\_BuildMode is created, which is used to act as an interface for the player to build the building in the Build Mode. The Player first selects the building he wishes to maker by using the number keys, the building is only selected if the player has enough supplies to create the building otherwise the building cannot be selected. The sprite follow the mouse position. When the mouse button is pressed to create the building, the objects checks if the place is empty after which the object is placed if the place is empty.

### Pausing of Game

* **Status:** Completed
* **Overview:** The player is able to pause the game and go to a pause menu, which then allows his to resume, restart or return to the main menu.
* **How it was implemented:** The game room is made persistent. A variable called restart is initialize in the obj\_Control object. In the step event, whenever, the restart variable is true, the room’s persistent value is set to false and the room\_restart() function is called to restart the game room. As for pause since the game room is persistent, the room is not deleted when the rooms are changed, hence, the player can navigate to pause room and come back and the game will still go on. As for main menu the room is just changed back to the main\_menu room.

### Random Spawning of Supply and Ammo Cache

* **Status:** Completed
* **Overview:** The Ammo and Supply caches spawn at the randomly around the map, which are required by the player. The ammo is used to provide the player with ammo while the supply is used to make buildings in build mode. The Supply cache also replenishes some of the player’s health.
* **How it was implemented:** A cache object is created, which spawns around the map at random avoiding the center of the room and staying at the edges of the room. If the Object collides with a tree or spawns outside the room or collides with another cache, the move\_random function is called in the step event. Hence making sure that all the caches are evenly spaced out throughout the room. The cache is created in the step event of the obj\_Control object, when the round starts.

### Shooting recoil

* **Status:** Completed
* **Overview:** The When the player shoots, the bullets do not travel exactly where the player’s gun is pointed, but has a recoil causing the bullet to deviate left or right. This is the accuracy the player has with the gun and can be upgraded in the upgrade menu of the game.
* **How it was implemented:** When the bullets are created in the step event of the player object, a random integer from a range of -10 to 10 is added to the direction of the bullet, which is equal to the direction of the player, in order to create the effect of a recoil occurring when the shots are fired.

# Student Name: Tan Yong Wei

### Animation for button

* **Status:** Completed
* **Overview:** Every button will have an animation of its own – normal, hovered and clicked.
* **How it was implemented:** Every button sprite will have 3 sub-images, normal, hovered and click all in order. A button object was then created with the relevant events – mouse over, left pressed and created, where upon creation, the image index would be the first, when mouse over would be the second and so forth.

### Animation For Player

* **Status:** Completed
* **Overview:** The player object would have various animations for moving and not in motion.
* **How it was implemented:** The player object would have three sub-images – one for not in motion and the other two for movement (left leg up and right leg up). Upon moving the player object, the player image index will be switched back and forth between the two movement sub-images and would change to the other when all movement is stopped.

### 8 Direction Movement

* **Status:** Completed
* **Overview:** The player object is able to freely move in all 8 directions (North, North-East, East, South-East, South, South-West, West, North-West).
* **How it was implemented:** Using the default controls (W, A, S, D), the player will be able move in all 4 directions. Upon holding more than one adjacent controls, the player will be able to move diagonally.

### Player Upgrade System

* **Status:** Completed
* **Overview:** The player will be able to upgrade certain abilities with an upgrade menu.
* **How it was implemented:** The variables used for upgrade will be moved to the control object which the player will use it as reference for these variables (such as player health). Through the pause menu, the player can select the upgrade menu and upgrade his or her desired abilities using the supplies picked up during the game.

### Leaderboard

* **Status:** Completed
* **Overview:** The player will be able to see the top 10 players of the game and their in game statistics.
* **How it was implemented:** Using an external .ini file, players will be able to save their statistics in the game if he or she managed to enter the leaderboard. Afterwards, everyone can view the top ten players of the game via the Leaderboard which will once again load the external .ini file for the records.

### Main Menu

* **Status:** Completed
* **Overview:** A functional interactive main menu was created for the player to navigate through the game.
* **How it was implemented:** The main menu consists of images and buttons. These buttons make use of the button control object which is used to assign the on click events for these buttons.

### Audio/Music

* **Status:** Completed
* **Overview:** Audio or music are assigned to selected events or actions of the game.
* **How it was implemented:** Royalty free music or sounds are downloaded and loaded into the game based on the selected events or actions.

# References