Adam Talbot

0707754

Assignment 1

Software Design for Ricochet

(CP2119) GAMES DEVELOPMENT 1

Contents

Introduction....3
Analysis....4
  Aims....4
  Assumptions....4
Initial Design....5
    Player....5
    NPC....6
    Object....7
    Menu....7
    Level....8
    Main....8
  Implementation tools....8
  Class Diagram....9
Final Design....10
  Menu....10
    Screen Designs....10
  Player....12
    Shield Physics....14
    Camera....15
  NPC....16
  SFX....17
  HCI Considerations....18
Testing Strategy....19
Time plan....23

Introduction

The aim of this report will be to produce a detailed document planning and designing the creation of a 3D third person shooter style game.

The plan for my program is to load a bug-free game whereby the user takes control of a player character and can interact with it in various ways. The setting of the game will be in a long corridor shaped environment with the player standing at one end and waves of enemies incoming from the other end. The style of the game might be compared to that of a space invaders type game but with major alterations visually and in game play.

The player will have the ability to shoot at the enemies with an unlimited supply of basic ammo, dealing damage to the enemy units until their HP drops to 0, and in turn must avoid incoming fire from enemy units and keep their own HP above 0 or its game over. As well as this the player will be able to pick up items off the floor, such as special ammo which has a limited supply but deals increased damage to enemies, health packs which will restore a percentage of the players total HP, and thirdly I plan to have a special shield type item. The shield item will give the player immunity to damage while it remains active, instead incoming fire will actually bounce off the shield changing direction and dealing damage back at the enemy unit. I plan for the shield not to simply return direction back at the enemy but to use a physics system so that depending on the position and angle the bullet hits the shield will determine the direction it bounces back, I want the deflected bullet to be able to continuously bounce off the walls in the room until it reaches the other end.

The enemy NPC's will come in groups of three standing in a row at the end of the room, all of which will display their own hp above their heads and fire randomly in straight lines down the room towards the player. There will be a reinforcement counter displayed on the screen showing how many enemies there are left until the level is complete, as you defeat an enemy if the counter is above zero another enemy will spawn. As you progress through the levels there will be more enemies for you to face, enemies will fire more often at the player and bullets will travel faster.

Once it's game over, weather the player has beaten all level or if the player dies a high score will be displayed based on the level achieved and the amount of enemies defeated.

Analysis

Aims

The aim of the program will be to setup a 3D room object for the environment, a character object and enemy NPC objects. These will need to have appropriate animations, SFX and the logic to react to difference circumstances to create a fun and interactive combat system.

Here are some of the key aims required;

- Keyboard input (possibly mouse also)
- Sprites for the menu and score screens
- A 3D object the player controls and can move left and right
- Third person camera setup based on player object current position
- Animated player object
- 3D object for room setting
- 3D objects for enemy NPC's with firing, moving & death animations
- Items which move across the floor towards the player which can be picked up
- Shield which reflects bullets back to enemy when hit, dealing damage to them
- Advanced shield features allowing bullets to bounce realistically over the room
- Collision with walls and items
- Ability for player to fire upon the enemy
- Randomized firing from enemies towards the pc
- A spawning system whereby enemy NPC's are spawned into the area taking the position of recently deceased NPC's and decrementing the reinforcement count
- Player and enemy units de-spawning upon 0 HP
- An in game GUI displaying statistics (i.e. HP, Ammo, Items, Score)
- Different levels of game play, increasing the difficulty the greater the level.
- Scoring system
- Sound effects
- Blocking fire on some enemy NPC's (such as bosses)

Assumptions

The following assumptions are made;

- The enemy AI will always know what position the pc (Player character) is at.
- The enemy will know if the pc is firing special ammo upon them.
- The enemy will know how long the pc's special shield has remaining before it expires

Initial Design

Based on the aims of the program I have decided on what classes I am going to have, there will be 6 in total including the main class, these are; Player, NPC, Level, Object, Menu and Main. Now let's decide what each class's purpose is and what is required from them;

Player

- **Controls** – Using WASD or arrow keys allow player to move the object around the environment.
- **Spawning** – Allow spawning of the player object based on X,Y,Z values entered
- **Animate** – Setup animations to be played appropriate to the objects current action (shooting, walking)
- **HP** – Set a max and current HP for the player allowing it to be increased by heal kits or decreased by incoming fire
- **Despawn** – despawn the player upon reaching 0 HP
- **Shooting** – allow player to fire in a straight line towards the enemy NPC in front of them displaying a bullet object from the players gun travelling towards the enemy, dealing damage and playing sounds appropriate when the bullet makes contact
- **Camera** – Setup a third person camera to stick behind the player object and move according to player movement
- **Items** – Allow the player to have a small inventory of items, a max of 5 heal kits, 100 special ammo and 1 special shield item
- **Shield** – Attach a shield model to the front of the player model upon picking up a shield, bullets colliding with the player will then instead bounce back according to the angle of collision and damage the enemy it collides with.

NPC (enemies)

- **Spawning** – Allow spawning of enemy NPC models, these will be positioned in 3 positions at the end of the room, left, middle and right. Once an enemy dies if the counter is greater than 0 after a short interval another enemy NPC will be spawned in the correct position, taking the place of the deceased NPC
- **Reinforcements** – Add a counter on screen displaying current reinforcements left (e.g. 10) this will be used to keep spawning NPC's once they die and to add to the players score (Giving a higher score the more enemies have been killed)
- **AI** – Setup different AI states for the enemy NPC's to be in, these will be;
  - **Idle** – Simply loop an idle animation and remain stationary
  - **Attacking** – Set random timers e.g. between 2 – 10 seconds, once the timer is up fire upon the player, the timer will then reset, the player then has a chance to move out the way of the incoming bullet in which case it will despawn else if it collides with the player object it will decrement their HP a given amount.
  - **Death** – Upon reaching 0 HP regardless of current state the NPC will be in the death state, this will play their dying animation plus any SFX, once this has finished after a certain time the object will despawn and another object will respawn in its place.
  - **Roam** (possibility) – Based on size of the room and objects and any possible constraints I may or may not implement the roaming state instead leaving them stationary in a row. If I decide during implementation that it will be suitable I will set it up like this; storing the starting spawn position I will use this as the final return position, to create a random roaming illusion I would set flags to allow the NPC to roam a maximum of 3 times by setting a random angle, moving them forward for 1-5 seconds, make them stationary for so many seconds then repeat, and on the fourth move force them to return to their spawning co-ordinates so as they don't wander off too far.
- **HP** – Set a hp value for each NPC, this value can be decreased by incoming fire when the pc's bullet object collides with the NPC, it will also perform a check to see if the incoming fire is from normal or special ammo, increasing the decrement to hp accordingly, once hp reaches 0 the NPC will die (load death state)
- **Animate** – Setup animations to be played appropriate to the objects current action (shooting, dying)
- **Difficulty** – this will start at 0 and increase with each level, the higher the difficulty the more enemies will be spawned, the faster they shoot, and the faster bullets move

Object

- **Spawning –** The main function of the object class, to spawn and position various objects based on the type and X Y Z coordinates given.
- **Rotate –** To rotate the initialized object a set amount to look realistically places in the environment
- **Animate –** Possible needed if any objects have animations that need to be played/looped
- **Despawn** – Remove and delete the specified object

Menu

- **Adding Sprites –** to be able to create sprites from images and position them according to coordinates given
- **GUI –** To decide which GUI screen to be loaded this will be between;
    - **Main menu –** The initial main menu screen containing a background image, and 3 sprite buttons, Play, Options and Exit
    - **Options –** The options screen which will contain a background and sprites for the different options settings e.g. turn SFX on/off
    - **Game GUI –** This will be the GUI screen to be displayed once the player has clicked play and is in the game, this will contain a bar on the right displaying Scores, inventory, ammo etc.
    - **Game Over –** This will be the screen to be displayed during the main game if the player either completes the game or dies, displaying the appropriate message, but showing the players score regardless of win/lose
- **Load images –** a void function that can be included when appropriate to load images, i.e. at the very start of the program or upon clicking play

Level

- **Level** – This will choose which level is to be loaded based on how many levels the player has completed, this will set the appropriate difficulty in the NPC class
- **Score** – This will be the players current score and will be displayed on the right of the screen atop the GUI, this will increase based on level achieved and enemies defeated

This class will also be where the objects, NPC's and menu screens are instantiated and spawned, this is an appropriate place to load and delete all models and sprites used in each level to keep things neat.

Main

This will contain the main game loop and decide which order to load models and sprites to stagger the loading, instantiate level and player when appropriate. Upon exiting the program it will delete all sprites, images & objects loaded.

Implementation tools

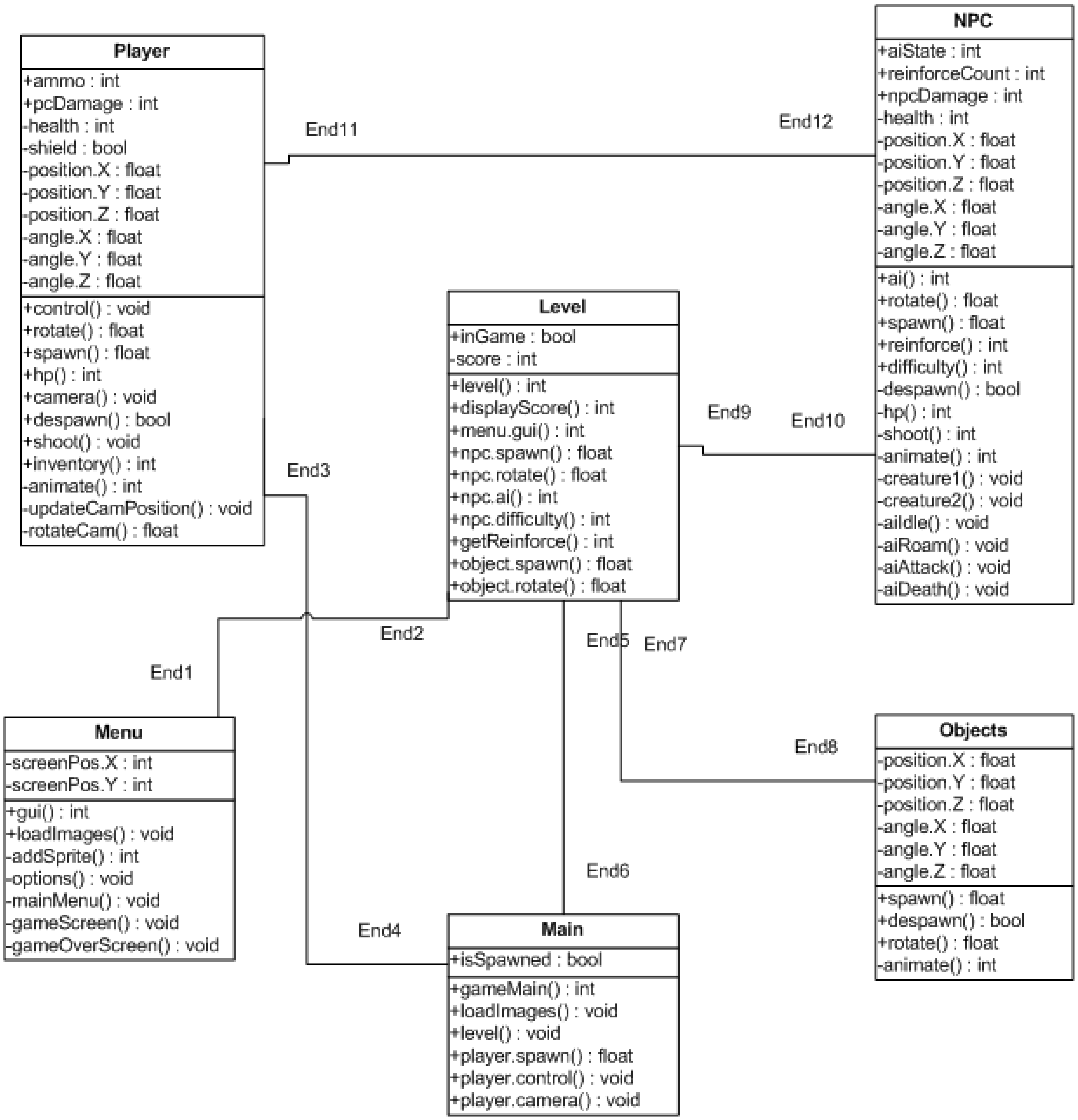
Software I will be using to make the program;

- Visual studio 2005
- DirectX 9.0
- Egg framework
- 3DS Max
- Paint

Visual Studio will be my main compiler, I am using 2005 as this is the version that best supports the egg framework, this is a small DirectX 9.0 wrap provided by the university to help speed up the process of basic tasks such as sprite and object loading, positioning and rotating. Paint will be used to draw any images and 3DS Max will be used to create any 3D models that I can't find royally free.

Class Diagram

This is a brief class diagram outlining the main attributes and operators each class will need and which classes will link together;



Final Design

This section will be expanding on areas from the initial design giving pseudo code and algorithms for the more advanced class functions.

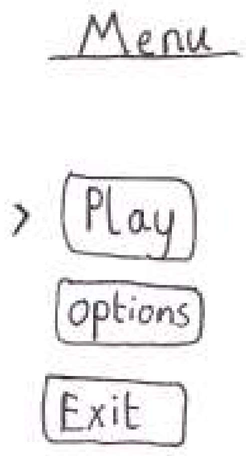
Menu

Screen Designs

The following is a rough idea of the layout of the game screens; this will be the role of the Menu class to display the appropriate GUI.

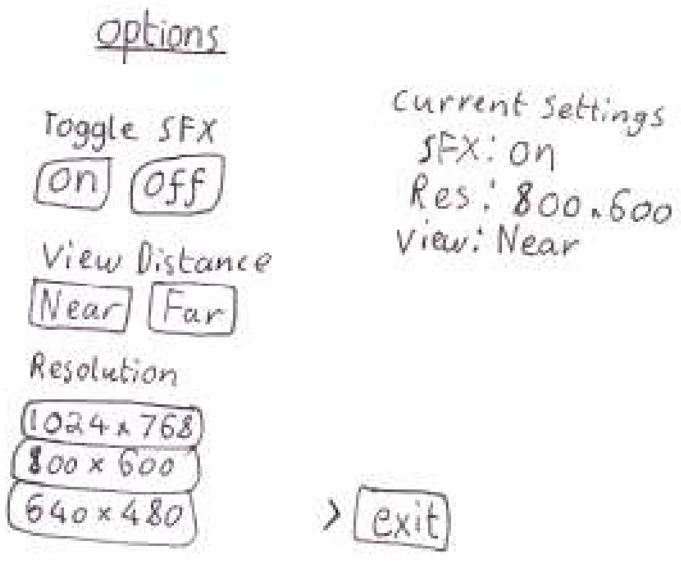
Menu GUI

This will consist of a simple sprite background and sprites on top representing the three buttons, play, options and exit. The player will use the keyboard arrow keys to move between the choices which will Display an arrow next to the button currently selected.



Implementing the selection will be simple, here's some brief pseudo code explaining it;

```
If button is 1
        Sprite arrow X,Y = buttonX-50,buttonY+10
If enter key is true
        If button is 1
                Load game
        Else if button is 2
                Load options
        Else
                Exit
If down key is 1 button++
If up key is 1 button - -
```

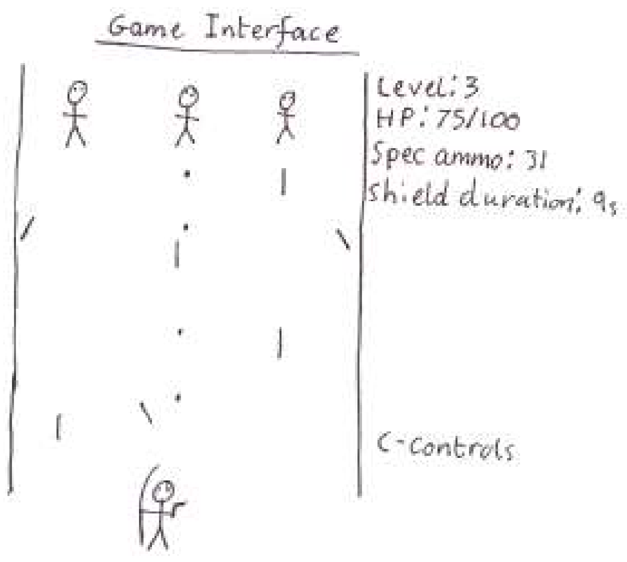


Options GUI

The options button will take you to a screen where you can change SFX resolution view controls etc, and will display the current setting in the top right of the screen.
The exit button in the bottom right will allow the user to go back to the menu screen.

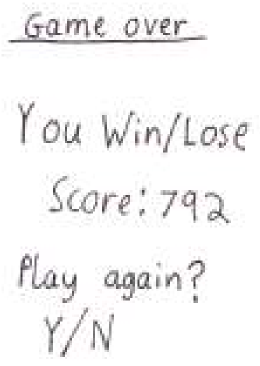
Game GUI

If the user selects play, the main game will then start, to avoid being thrown straight into battle there will be a countdown on screen from 3 – 0 to get your bearings. In this phase the menu sprites and background well all be hidden and the in game GUI will be loaded, with the main models of the game being on the left and the sprite display and information on the right. This will display the current game level, the player's hp min/max, the amount of special ammo they have left and shield duration remaining (if shield is picked up).



Game over GUI

The final screen will be the game over screen, this will be displayed once the player has completed all 5 levels or the player has died, displaying 'You Win' or 'You Lose' appropriately, it will also display the players score and give the user a choice to play again or exit the program with Y/N keys.



Player

Let’s decide how to implement the main features of the player class from the initial design.

For control I plan to have the player constantly facing the end of the room towards the enemies, except when running from side to side then it will face the way it’s running and snap back to facing the enemies when stopped.

Here a basic idea of how I will set up the controls and functions;

```
If left key is pressed
        Player.moveleft ()
If Right key is pressed
        Player.moveright ()
If Space key is pressed
        Player.shoot ()
```

Movement of the character which will rotate the object, move & play its animation, here’s an example of one movement direction;

```
Moveleft (){
Rotate objectY -90
Object positionX - -
Sidestep anim();
If object player collides with wall
        Position.x = oldX // player stops at positon from previous cycle
        Position.z = oldZ
Else
        Update Position.x & position.z}
```

I will store the animations in a switch statement so they can be played easily

e.g.;

```
Switch (int animLoop)
Case 'shoot'
        Objectanimspeed (40)
        Loopobject animframe (1-45)
Case 'walk'
        Objectanimspeed (40)
        Loopobjectanimframe (46-66)
```

For shooting I will give the player an unlimited supply of basic ammo which will do a small amount of damage to the enemy's hp that's in front of the player when firing.

Once the player picks up special ammo their normal bullets will do double regular damage but as they shoot a counter of their current ammo will decrease, when it reaches 0 the player will return to normal damage.

An example to implement firing;

```
shoot (){
position objectBULLETXYZ = objectPLAYERXYZ
// check if player is on left, right or centre, then get co-ordinates of opposite enemy e.g.
if playerSIDE is left
targetX = npcLeftX, targetY = npcLeftY targetZ = npcLeftZ,
//get distance between target position and bullets spawn position and divide by a number to make a move speed, e.g. divided by 1000 means it would take 1000 code cycles of the bullet moving at that speed to reach its target  then multiply speed by a velocity value to make it travel faster
Velocity = 4
moveX = targetX – objectBULLETX / 1000 * Velocity
moveY = targetY – objectBULLETY / 1000 * Velocity
moveZ= targetZ – objectBULLETZ / 1000 * Velocity
```

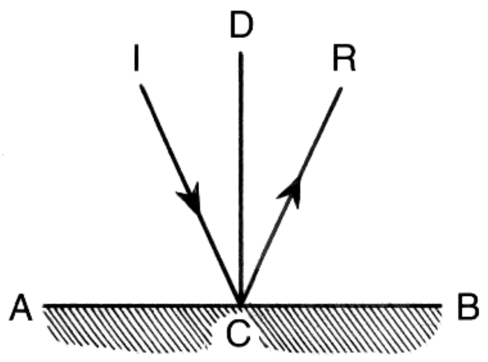
Once the bullet reaches its destination decrement the NPC's HP which resides in the spot fired at based on bullet damage.

Shield Physics

If the player picks up a shield by moving into the objects co-ordinates on the screen I will hide the shield object and position a shield on the player, if the player then moves into the line of fire from an enemy bullet the bullet will hit the player but do no damage, and instead will return a bullet back to the enemy it came from by working out a bounce angle to make the bullet bounce off the shield back to the enemies or into the walls of the room object.

Now let's think about the actual physics to make angled bouncing;

As the players shield will be in a crescent shape in front of the player when a bullet hits the shield based on the distance from the centre will determine how much to rotate the directional vector of the bullet in degrees. Then there's the issue of getting the bouncing angle of the bullet when it collides with the wall. In 2D this wouldn't be too hard to work out you could do it with trigonometry using math.sin and math.cos or another way would be simply to use the velocities horizontally and vertically (X,Y) and if the object collides with the top or bottom of the screen invert the Y velocity, if it hits the left or right invert the X velocity.



Angle of reflection by Eric Weisstein's World of Physics

This is a diagram I found which shows the kind of idea; whatever angle the object is at the time of collision will cause a mirrored returning angle on the other side.

I could work out a similar approach with 3D co-ordinates not making any changes to the Y co-ordinate and just altering the x & z values to make life simpler, as the bullets will all be travelling at the same distance from the ground all the time anyway, this would create a plausible effect. I feel as long as it looks reasonably realistic then it's a success, to create a perfectly accurate physics system would be a very time consuming task without the use of a physics library or much more advanced physics knowledge than my own.

Camera

For the Camera setup to give an accurate third person View I will use trigonometry. To set this up we need to know the horizontal rotation, vertical rotation and radius. The camera will rotate around the object as if in a sphere so to get the horizontal and vertical rotation we use the position of the camera on the circle around the object(horizontal) and the circle perpendicular to that(vertical). Firstly to get the Y rotation of the camera from the player's object (Working out the Y axis first as the rotations on this axis can alter the X and Z coordinates slightly);

Radius = constant distance from the object

vRot = vertical rotation (current X,Y,Z position of camera on vertical circle)

hRot = horizontal rotation (current X,Y,Z position of camera on horizontal circle)

*y1 = radius*sin(vRot) r1 = radius*cos(vRot)*

y1 = y component of camera, r1 = new radius to use

Radius has changed because since looking from the top down the camera will have moved closer to the centre (pc object) because of its movement vertically along the circle.

Now to find the X and Z components

*x1 = r1*cos(hRot) z1 = r1*sin(hRot)*

Now we have the x y and z components we can update the position of the camera;

```
UpdateCamPosition() {

 // radius is the r1 temporary radius stored earlier

position.X = (x1)(radius * Math.Cos(vRotation) * Math.Cos(hRotation));

 position.Y = (y1)(radius * Math.Sin(vRotation));

 position.Z = (z1)(radius * Math.Cos(vRotation) * Math.Sin(hRotation));}
```

To rotate the position of the camera, a simple function like this would work;

```
RotateCam(float h, float v) { hRotation += h; vRotation += v; UpdatePosition(); }
```

Using a function like this will allow us to zoom the camera based on distance provided;

```
Zoom(float dist) {

 radius += dist; if(radius < .01f) radius = .01f; UpdateCamPosition(); }
```

NPC

The NPC class will need to be able to load one or two different models (a regular enemy & a boss type) it will then need to position 3 models at the end of the room at a time, and when one enemy has been defeated, remove the model and replace it with another enemy model until the enemy reinforcement counter reaches 0. The enemy needs to be able to fire randomly in the player's direction, which will display a laser or bullet object and move it towards the player using, if it collides with the player it will damage the players max hp. If the player's hp reaches 0 the game will end and the game over screen will load.

The first thing in the enemy class will be to set up a row of 3 enemies, animate them, and begin firing.

Enemies on the field will be numbered 1 2 and 3 depending on their position. The numbers will decide the co-ordinates that the enemy object be positioned. If an enemy at position 2 dies for example I will set a timer to wait a few seconds then respawn another enemy at position 2's co-ordinates and number it e.g.

//upon death of an NPC in the death state store its position (left right centre) and set a timer for the new NPC to be spawned (spawnTimer)

```
if reinforceTimer > 0
    if spawnTimer > currentTime
        if emptySpace = 1 // 0-left 1-centre 2-right
        grunt.spawn('1',emptySpaceX[1], emptySpaceY[1], emptySpaceZ[1])
        reinforceCount - -
```

Firing will be fairly straight forward, enemies will have randomised timers so they fire bullet object which will travel in a straight line down the screen towards the player and hide the object once it's out of view, the higher the game level the more frequent I will make enemies fire and the faster the bullets will actually travel. HP decrement and bullet logic will be the same as discussed in the player class.

SFX

I plan to implement various music and SFX for the program; I will have a constant ambient sound looping in the background, and change to a different more battle orientation sound while the player is in the game state.

I will use single play sounds for all the different game effects, these I will find royally free or record myself, sounds I will need are;

- PC Gunfire – Regular ammo fire, and a different sound for special ammo fire
- PC Hit – Colliding sound of an enemy bullet damaging the player
- PC Move – Footstep sounds for moving the player around
- PC Pick up item – A sound to be played when an item on the ground is picked up
- PC Shield Deflect Sound – A sound of bullets being deflected off the PC's Shield
- PC Heal – A sound of the player using a healing kit
- PC Die – A sound of the player's death
- NPC Hit – A sound of enemy NPC's taking damage from the players gunfire
- NPC Gunfire – Sound of enemy gunfire
- NPC Death – A sound of enemies dying
- NPC Spawn – A sound each time an enemy is spawned onto the field
- Menu – A button click sound
- Menu – Button Highlight sound
- Menu – Game Over sound Win/Lose
- Ambience – Menu ambience
- Ambience – In game ambience

HCI Considerations

The aim of my design is to try and make interaction with the program as simple as possible for the user, giving clear names of choices and making button layouts and colours consistent on each screen. With HCI in mind I also need controls to be clearly listed, even with the controls listed in the options page users could still forget them by the time they go to play the game and not realise how to do stuff such as drink a potion or use a shield.

A solution could be to use keyboard controls while in the game view and have a shortcut key (e.g. F1) toggle an on-screen display of the controls.

I also need a clear way for the user to return to the menu page from the game screen, again I could do this with another keyboard key and have it request the user to then press 'Y/N' to confirm the choice, this will prevent accidental pressing of the key.

Testing Strategy

There will be a lot of testing required for this, in the following testing I plan to test each section in many ways

1. Sprite Testing – Ensure all Sprites display correctly over all the relevant sprites, and disappear when expected.
2. Button Testing – Ensure all buttons go to the correct screens and back again, make sure buttons still work when returning to the menu after closing the game screen. Make sure the co-ordinates properly cover the buttons for mouse click (e.g. You can't click outside the button to control it)
3. Boundary Testing – Make sure the player cannot go outside the map by trying to pass through as many points as necessary.
4. File Input – Check to see if it loads the values correctly, add spaces in the file change the numbers around, make sure it can load one file and then after returning to the menu load another file successfully.
5. Shooting – Make sure incoming and outgoing fire damages and collides with the appropriate target accurately.
6. Animations – Test to see if all animations play properly and loop when needed. Ensure appropriate animations don't run when there not meant to e.g. the running on the spot.
7. Camera – Does the camera follow the player properly, does it go through the walls or ground at any point.
8. Text – Check to make sure all on screen text displays properly over the appropriate screen and disappears when on different screens.
9. Crash Testing – Check to make sure you cannot crash the program, load files repeatedly, move the player into the walls while shooting and performing many actions at once, does any unusual behaviour occur?

Restarting – When returning to the menu check to make sure all settings are reset and all appropriate objects are deleted and reloaded, e.g. is the player in the correct starting position.

Player	Does it work	Problems encountered
Does player collide properly with walls		
Does player collide with bullets		
Does movement work		
Does player animations work		
Does player SFX work		
Does player accurately take damage		
Does player die on 0 HP		
Does player Pick up items accurately		
Does player use and decrement item counts accurately		
Does shield deflect bullets properly		

Objects	Does it work	Problems encountered
Do object models load correctly		
Are object using the correct texture		
Are object correctly positioned		
Are object correctly rotates		
Any other visual problems with objects on screen		

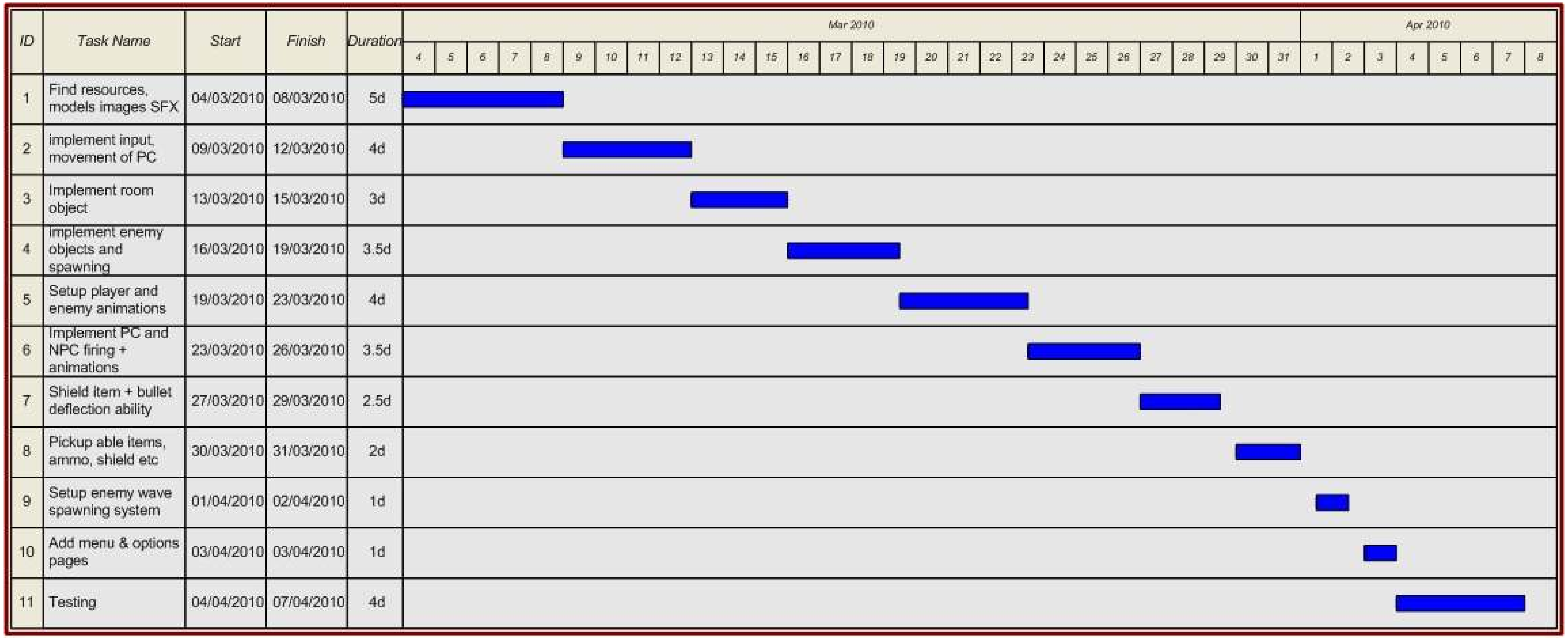
Enemy	Does it work	Problems encountered
Does Enemy Fire randomly		
Does Enemy collide with bullets		
Does movement work		
Does enemy animations work		
Does enemy SFX work		
Does enemy accurately take damage		
Does enemy die on 0 HP		
Does enemy despawn correctly and decrement reinforcements		
Does enemy respawn in correct positions		

Menu	Does it work	Problems encountered
Does Play, Option & Exit work from main screen		
Does SFX options work in options page		
Are all sprites and text displayed correctly		
Can you return to menu from in game screen		
Does High Scores work correctly		
Does Game over screen work correctly		
Do in game GUI images		

Level	**Does it work**	**Problems encountered**
Does the correct level load		
Does high scores display properly		
Is player & enemy models positioned correctly		
Do items display and move correctly		
Do enemies stop spawning when counter reaches 0		
display correctly		

Time plan

As I have six week to implement the program I have split the major tasks up into sections and set a rough time which I feel I can complete them in, this is as follows;



ID	Task Name	Start	Finish	Duration
1	Find resources, models images SFX	04/03/2010	08/03/2010	5d
2	implement input, movement of PC	09/03/2010	12/03/2010	4d
3	Implement room object	13/03/2010	15/03/2010	3d
4	implement enemy objects and spawning	16/03/2010	19/03/2010	3.5d
5	Setup player and enemy animations	19/03/2010	23/03/2010	4d
6	Implement PC and NPC firing + animations	23/03/2010	26/03/2010	3.5d
7	Shield item + bullet deflection ability	27/03/2010	29/03/2010	2.5d
8	Pickup able items, ammo, shield etc	30/03/2010	31/03/2010	2d
9	Setup enemy wave spawning system	01/04/2010	02/04/2010	1d
10	Add menu & options pages	03/04/2010	03/04/2010	1d
11	Testing	04/04/2010	07/04/2010	4d