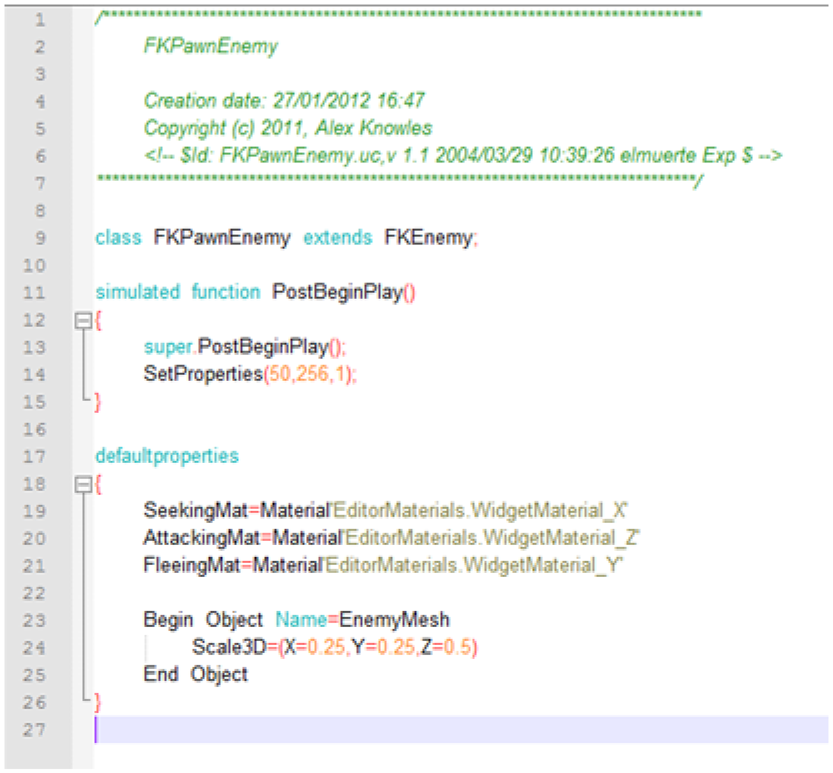
27/01/2012
Today I am getting rid of TestEnemy.uc to create a main enemy class in which each type of enemy will be derived from. This file will be called FKEnemy.uc and contain all base traits that all enemies will need.

For the moment I am going to keep the all the code from TestEnemy and experiment to find out what I will need to remove to get different enemy types to work. I have actually commented out the code that calls the set properties function from TestGame.uc, to allow the classes to be able to have coded in properties. I will come back to this later and allow the level designer to choose what type of enemy they want.

The first sub class I am making from FKEnemy is FKPawnEnemy. This will be the very basic enemy type that will have low life and normal speed. Not much code in this class currently:

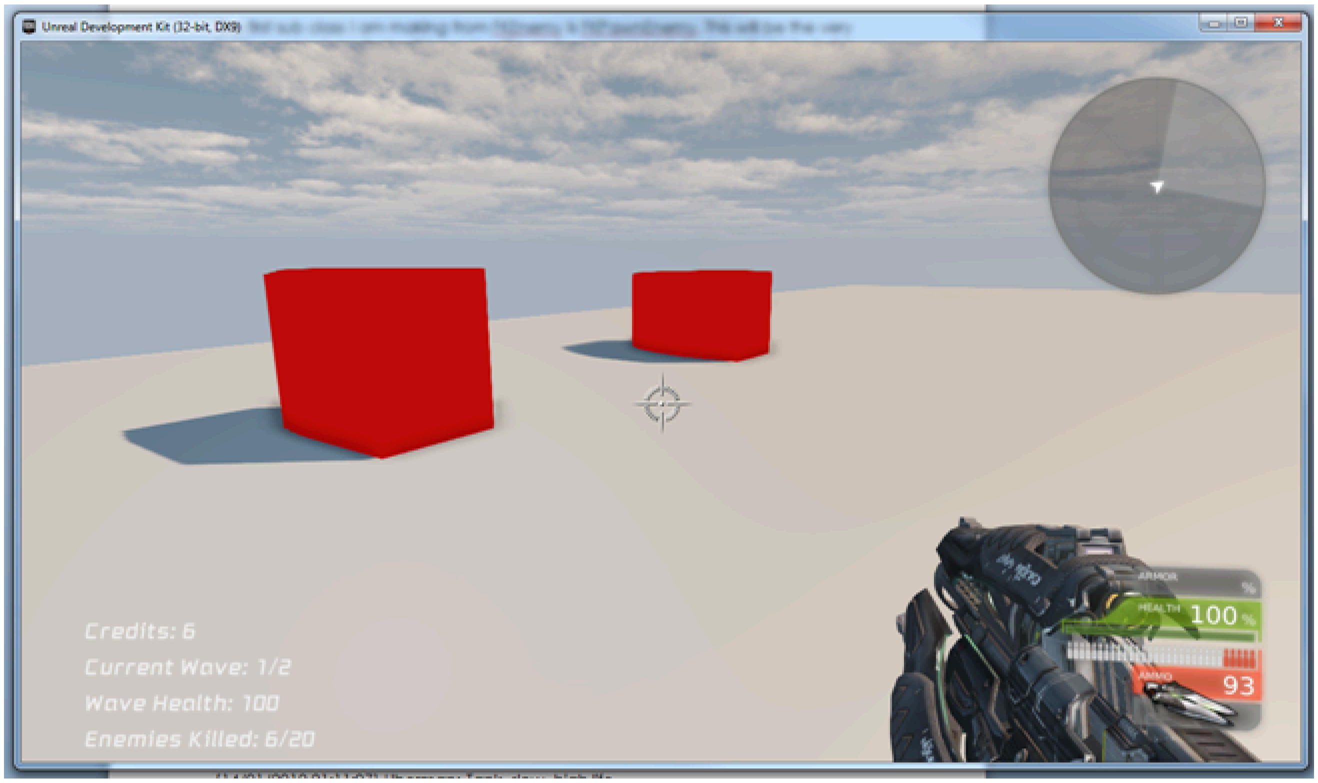


```
/******************************************************************************
	FKPawnEnemy

	Creation date: 27/01/2012 16:47
	Copyright (c) 2011, Alex Knowles
	<!-- $Id: FKPawnEnemy.uc,v 1.1 2004/03/29 10:39:26 elmuerte Exp $ -->
******************************************************************************/

class FKPawnEnemy extends FKEnemy;

simulated function PostBeginPlay()
{
	super.PostBeginPlay();
	SetProperties(50,256,1);
}

defaultproperties
{
	SeekingMat=Material'EditorMaterials.WidgetMaterial_X'
	AttackingMat=Material'EditorMaterials.WidgetMaterial_Z'
	FleeingMat=Material'EditorMaterials.WidgetMaterial_Y'

	Begin Object Name=EnemyMesh
		Scale3D=(X=0.25,Y=0.25,Z=0.5)
	End Object
}
```

All this code does is set the properties of the enemy, the texture and the size. On build of the code no syntax errors have been found so onto running the game.

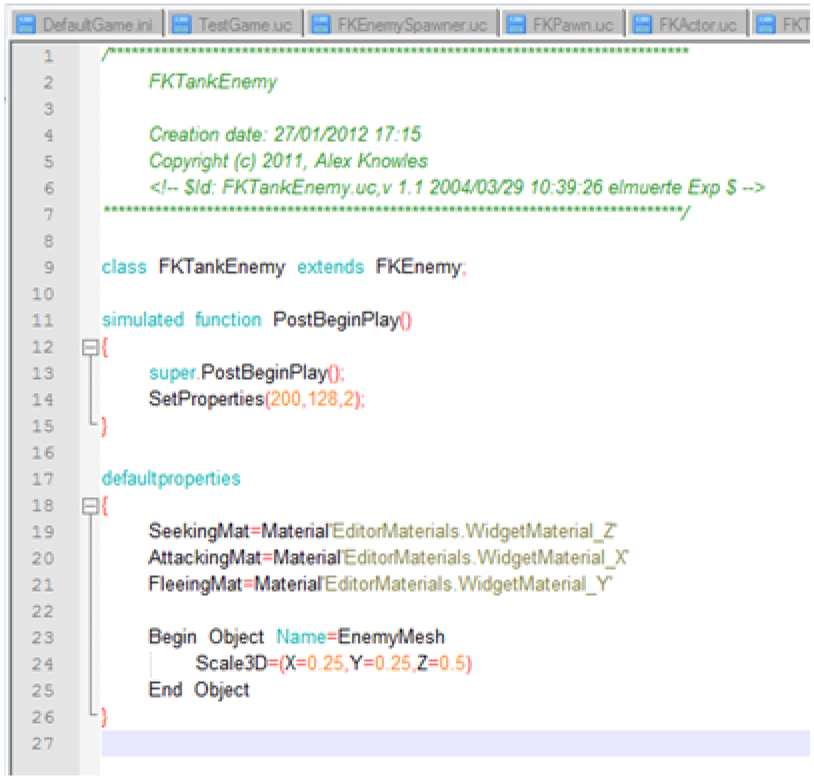
On running the game no enemies actually spawn but the counter for number of enemies killed keeps counting up.

Thinking about it I haven't changed what the enemy spawners are spawning so they're still spawning in TestEnemy.uc . I think I should change this to FKPawnEnemy.uc to see if what I have written works.



I was right! But annoyingly they move too slowly, that’s probably because I’ve set their speed to ten. I’ll amend that next.

Anyway, that was simple. Next up is the Tank enemy type. This enemy type is slow but has a lot of health. To set this up is pretty much the same as FKPawnEnemy but with a different size and texture. For the texture I’m just setting it backwards to the pawn, so blue when moving and red when attacking. This class will be called FKTankEnemy.

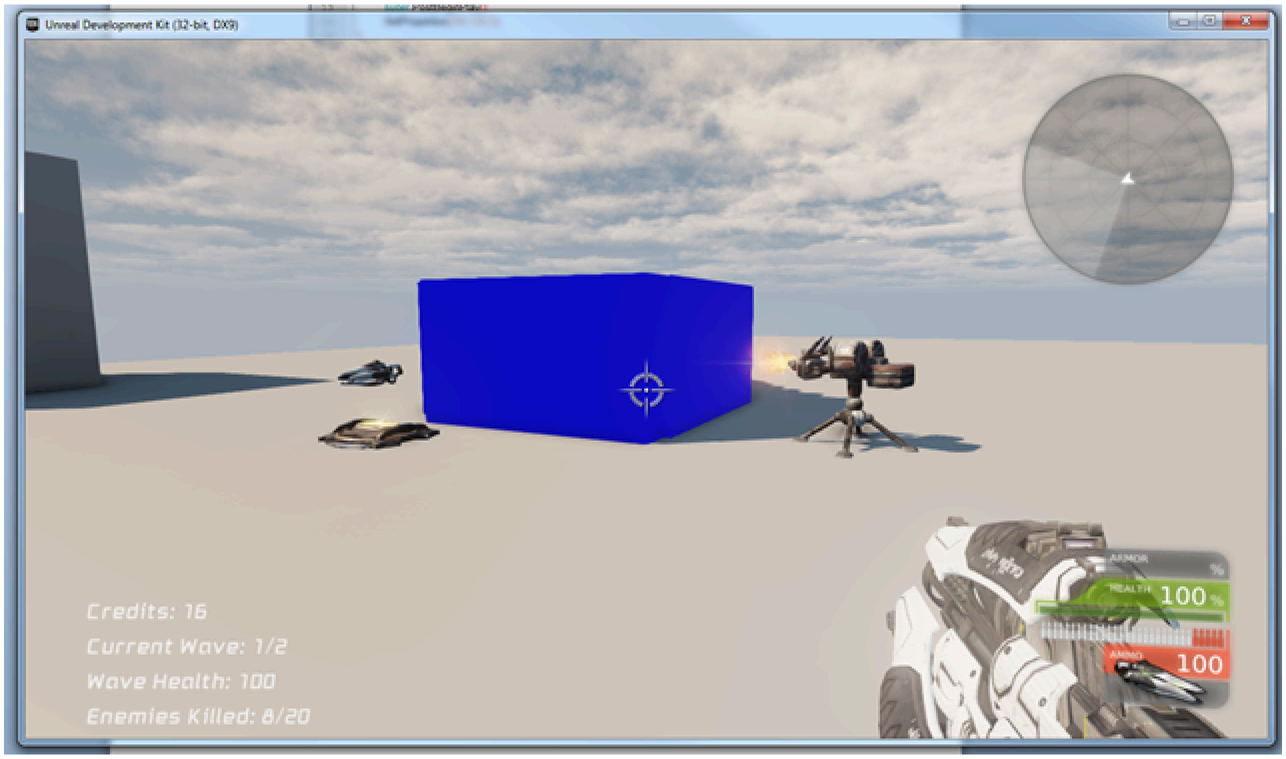


DefaultGame.ini | TestGame.uc | FKEnemySpawner.uc | FKPawn.uc | FKActor.uc | FKT

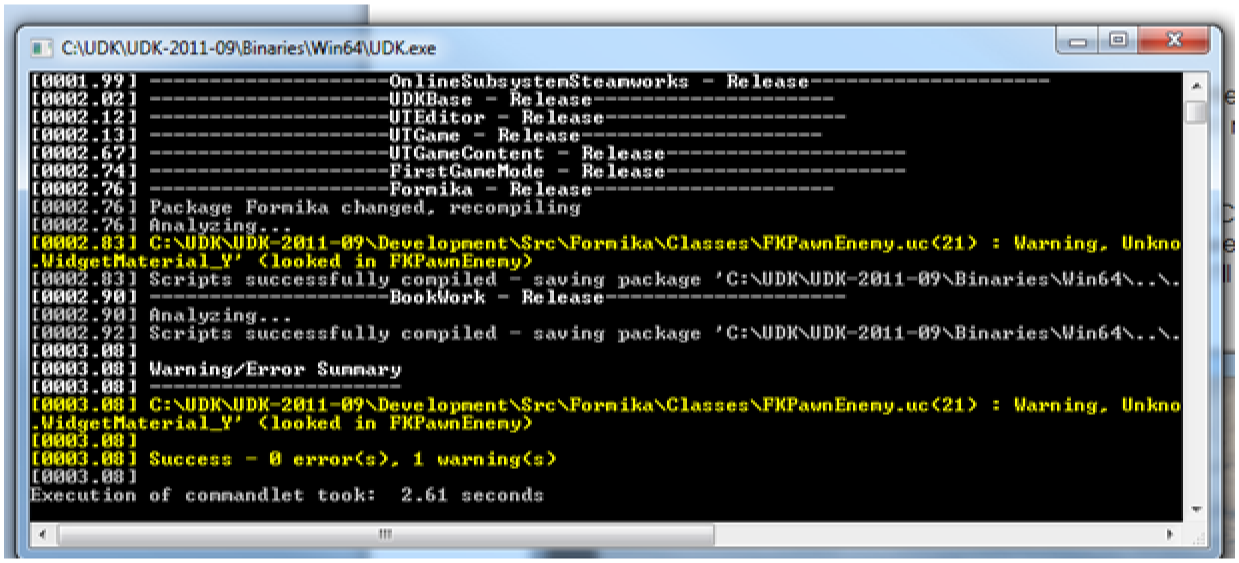
```
/*******************************************************************************
	FKTankEnemy

	Creation date: 27/01/2012 17:15
	Copyright (c) 2011, Alex Knowles
	<!-- $Id: FKTankEnemy.uc,v 1.1 2004/03/29 10:39:26 elmuerte Exp $ -->
*******************************************************************************/

class FKTankEnemy extends FKEnemy;

simulated function PostBeginPlay()
{
	super.PostBeginPlay();
	SetProperties(200,128,2);
}

defaultproperties
{
	SeekingMat=Material'EditorMaterials.WidgetMaterial_Z'
	AttackingMat=Material'EditorMaterials.WidgetMaterial_X'
	FleeingMat=Material'EditorMaterials.WidgetMaterial_Y'

	Begin Object Name=EnemyMesh
		Scale3D=(X=0.25,Y=0.25,Z=0.5)
	End Object
}
```

Pretty much the same as before but it has different properties and the attacking and seeking materials swapped. One more thing before I build I must change the spawners again so they spawn tanks instead of pawns.

Build was successful. And the enemies spawned correctly. Could possibly do with a bigger gap between spawning and I also saw that the tower isn't shooting. I haven't change TestEnemy to FKEnemy so the towers won't work, I'll change that now, shouldn't really bring up any problems.



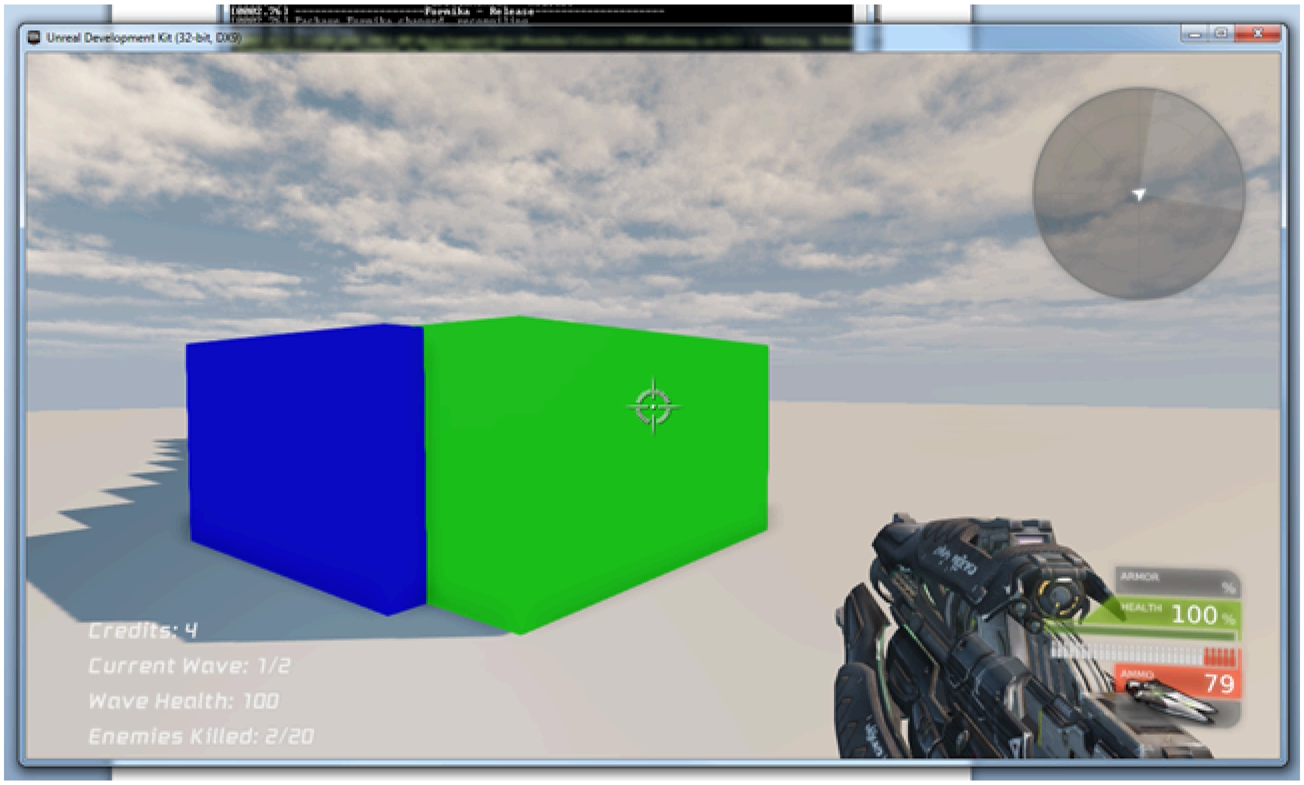
Just had a thought, I want to see when the enemy takes damage so seeing as I don't use the fleeing material I'll just use that. This just involves changing the variable storing the material from FleeingMat to DamageMat and then setup a half a second timer which will change the material.



C:\UDK\UDK-2011-09\Binaries\Win64\UDK.exe

```
[0001.99] --------------------OnlineSubsystemSteamworks - Release--------------------
[0002.02] --------------------UDKBase - Release--------------------
[0002.12] --------------------UTEditor - Release--------------------
[0002.13] --------------------UTGame - Release--------------------
[0002.67] --------------------UTGameContent - Release--------------------
[0002.74] --------------------FirstGameMode - Release--------------------
[0002.76] --------------------Formika - Release--------------------
[0002.76] Package Formika changed, recompiling
[0002.76] Analyzing...
[0002.83] C:\UDK\UDK-2011-09\Development\Src\Formika\Classes\FKPawnEnemy.uc(21) : Warning, Unkno
.WidgetMaterial_Y' (looked in FKPawnEnemy)
[0002.83] Scripts successfully compiled - saving package 'C:\UDK\UDK-2011-09\Binaries\Win64\..\.
[0002.90] --------------------BookWork - Release--------------------
[0002.90] Analyzing...
[0002.92] Scripts successfully compiled - saving package 'C:\UDK\UDK-2011-09\Binaries\Win64\..\.
[0003.08]
[0003.08] Warning/Error Summary
[0003.08] ---------------------
[0003.08] C:\UDK\UDK-2011-09\Development\Src\Formika\Classes\FKPawnEnemy.uc(21) : Warning, Unkno
.WidgetMaterial_Y' (looked in FKPawnEnemy)
[0003.08]
[0003.08] Success - 0 error(s), 1 warning(s)
[0003.08]
Execution of commandlet took:  2.61 seconds
```

On compiling the code I got a warning sayin that SeekingMat doesn't exist in the parent class so I need to remove it from FKPawnEnemy and FKTankEnemy.



Everything is looking fresh :P Only thing is that half a second is just a little too long so I'm going to try 1/4 . Still not right I'm going to try 1/8. Right, that's cool.

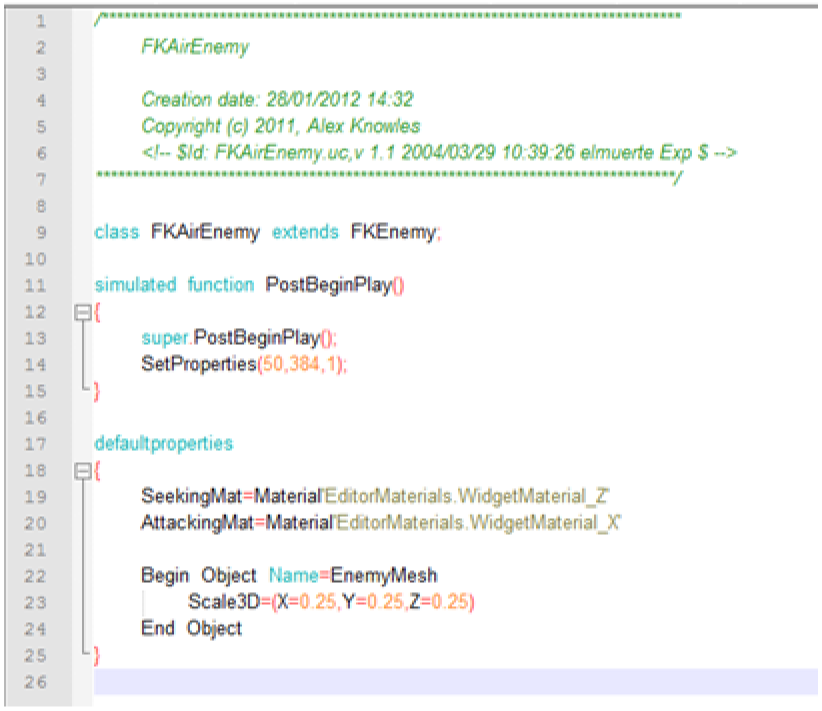
The next enemy type is an air unit. This one will need more coding as the enemy should never touch the ground but we still want it to like swoop down and damage the player.

I'm thinking this will work by increase the z position of the enemy when they spawn.

28/01/2012

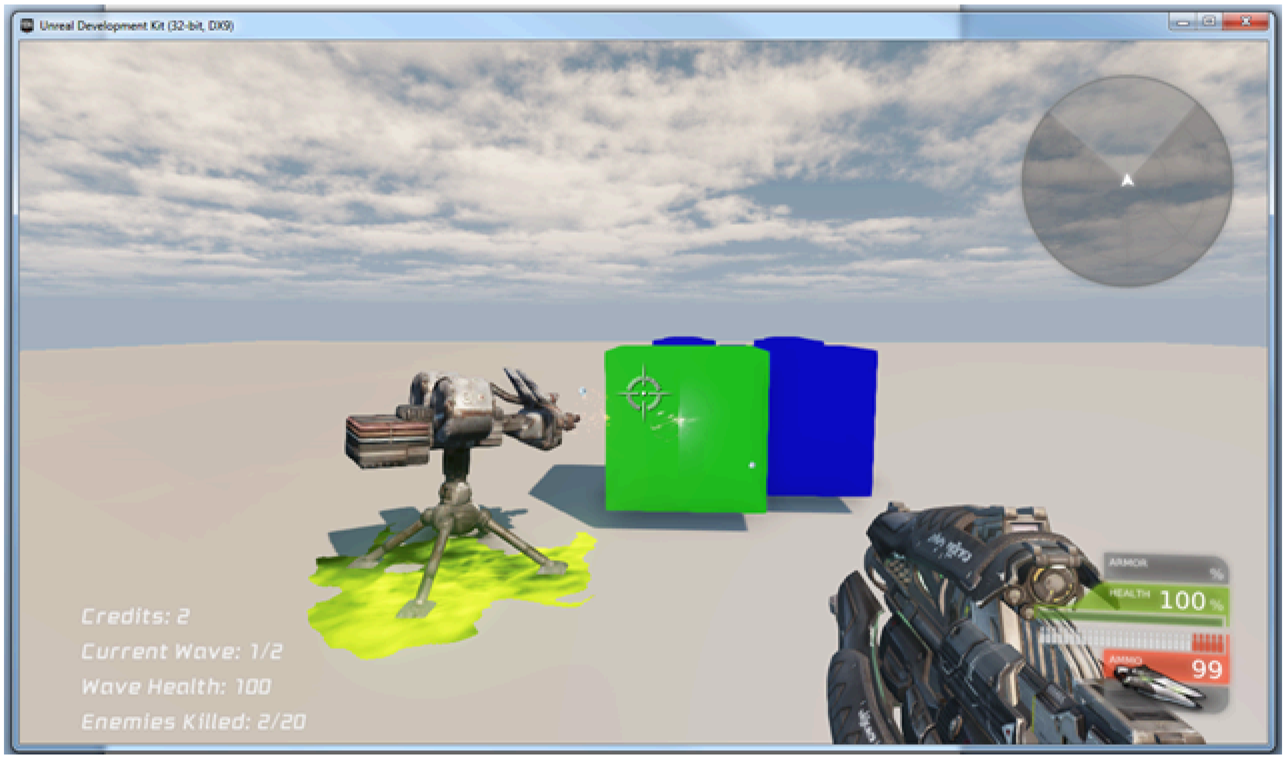
Going with what I said yesterday I'm going to play round with its spawning. I probably should break down what I said the enemy should do and produce it in code but this way is more fun :D

First off I'm going to copy the code from either FKPawnEnemy or FKTankEnemy into a new file and call it FKAirEnemy.uc. About the colour, I'm going to keep it the same as the pawn enemy. The size of the Air enemy will be different to both Pawn and Tank Enemy, I'm thinking smaller with less health but faster than the other enemies.



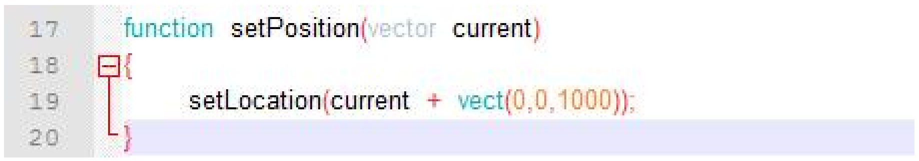
```
/*******************************************************************************
	FKAirEnemy

	Creation date: 28/01/2012 14:32
	Copyright (c) 2011, Alex Knowles
	<!-- $Id: FKAirEnemy.uc,v 1.1 2004/03/29 10:39:26 elmuerte Exp $ -->
*******************************************************************************/

class FKAirEnemy extends FKEnemy;

simulated function PostBeginPlay()
{
	super.PostBeginPlay();
	SetProperties(50,384,1);
}

defaultproperties
{
	SeekingMat=Material'EditorMaterials.WidgetMaterial_Z'
	AttackingMat=Material'EditorMaterials.WidgetMaterial_X'

	Begin Object Name=EnemyMesh
		Scale3D=(X=0.25,Y=0.25,Z=0.25)
	End Object
}
```

After coping and editing slightly I'm just going to check that the code works by getting them spawned in game.

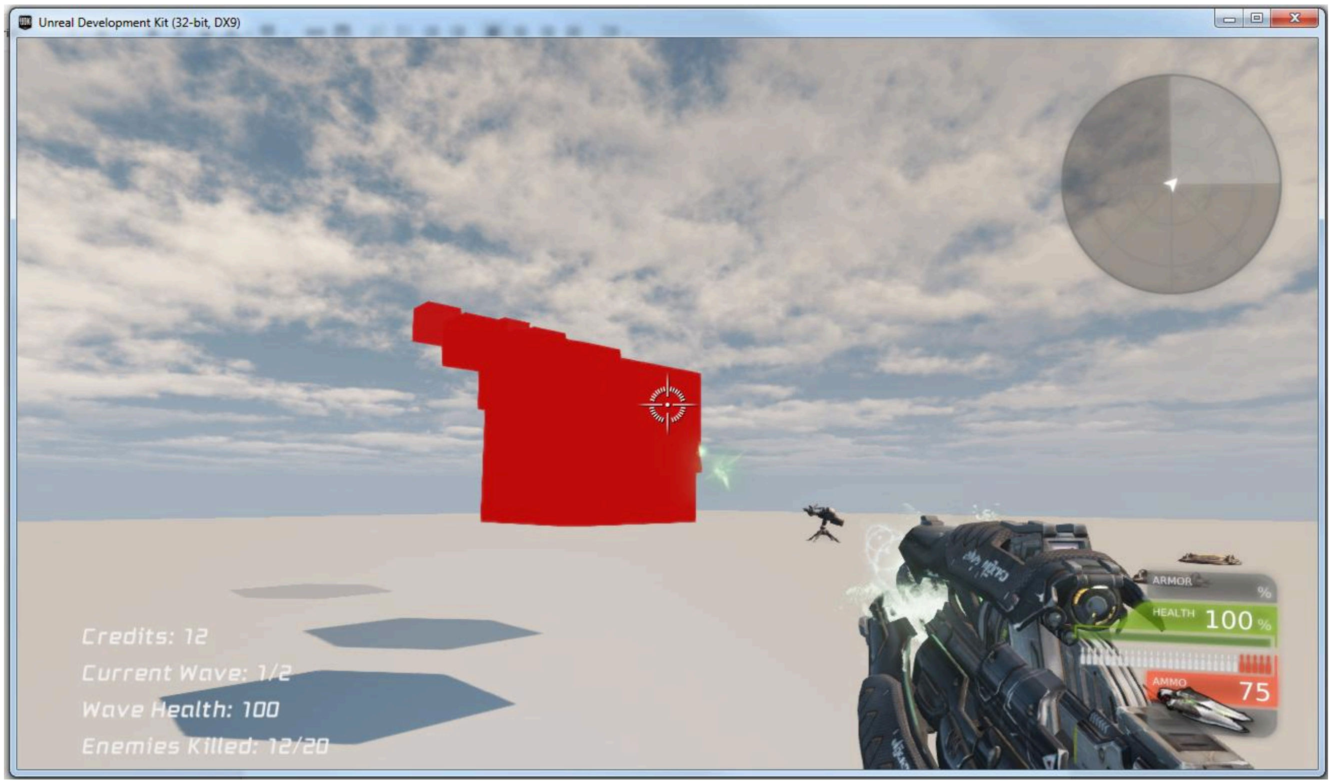


The build was fine and the enemies seem to work and I'm happy with their size.

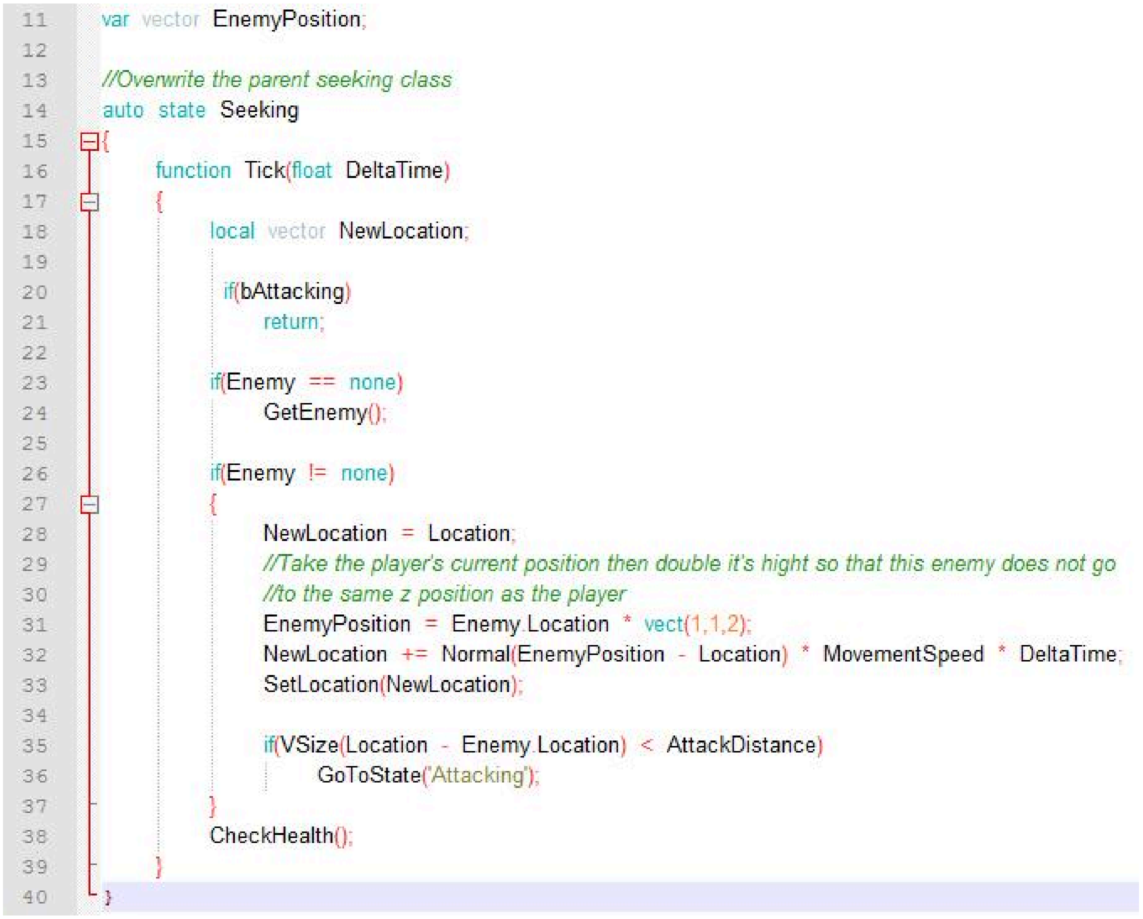
03/02/2012
Now that my computer is working I can carry on making the air enemy work. So to start with i am going to do what i said before; which was to move the enemies higher up when the spawn. This just involves a new function with one line of code in it. This can then be referenced when the enemy is spawned sending the spawners location as well.



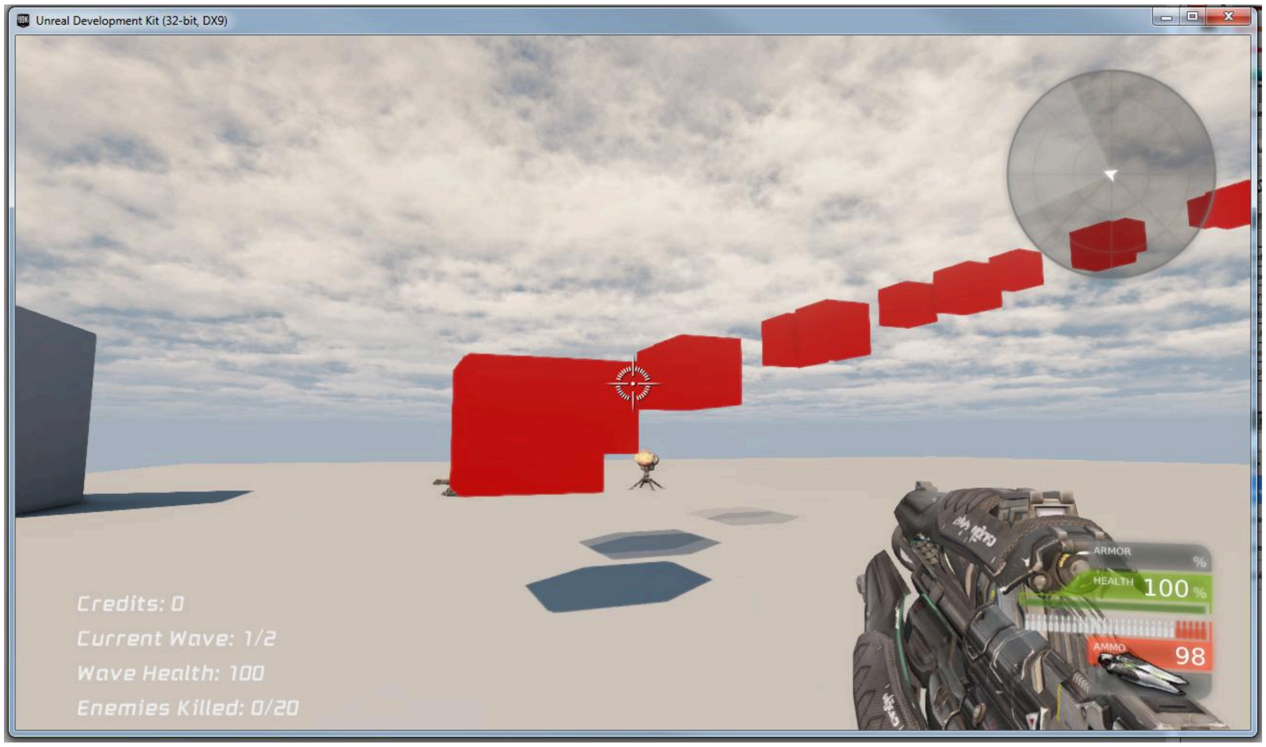
```
function  setPosition(vector  current)
{
      setLocation(current  +  vect(0,0,1000));
}
```



The build was successful and the enemies spawned 1000 higher that the spawner but the kind of float down towards the player until they're the same height. I going to change that so they never go lower than the players head. To do this I'm thinking that i need to overwrite the Seeking state and then make the enemy head for double the players hight.

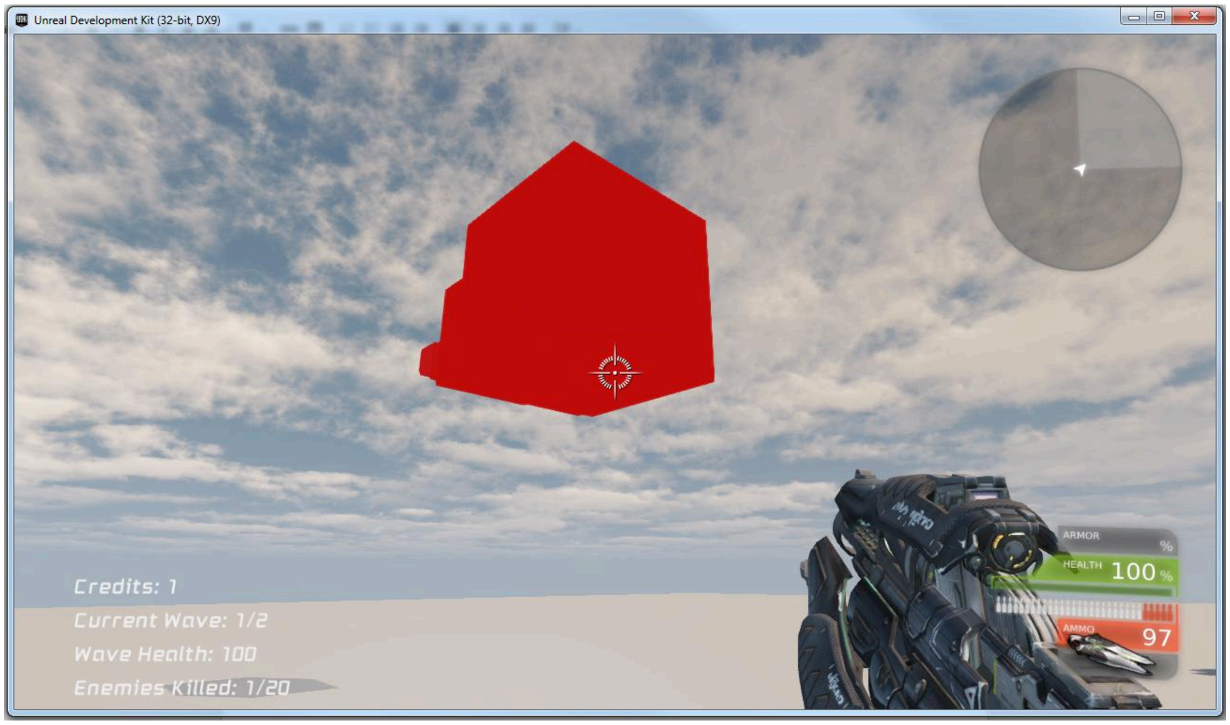


```
var vector EnemyPosition;

//Overwrite the parent seeking class
auto state Seeking
{
    function Tick(float DeltaTime)
    {
        local vector NewLocation;

         if(bAttacking)
            return;

        if(Enemy == none)
            GetEnemy();

        if(Enemy != none)
        {
            NewLocation = Location;
            //Take the player's current position then double it's hight so that this enemy does not go
            //to the same z position as the player
            EnemyPosition = Enemy.Location * vect(1,1,2);
            NewLocation += Normal(EnemyPosition - Location) * MovementSpeed * DeltaTime;
            SetLocation(NewLocation);

            if(VSize(Location - Enemy.Location) < AttackDistance)
                GoToState('Attacking');
        }
        CheckHealth();
    }
}
```

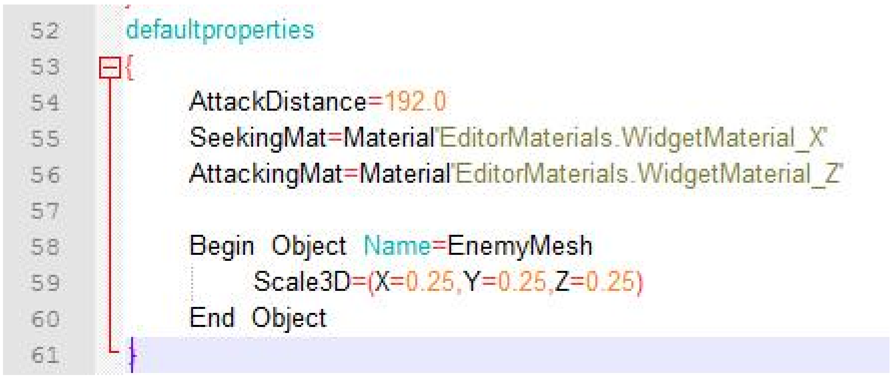


Credits: 0
Current Wave: 1/2
Wave Health: 100
Enemies Killed: 0/20

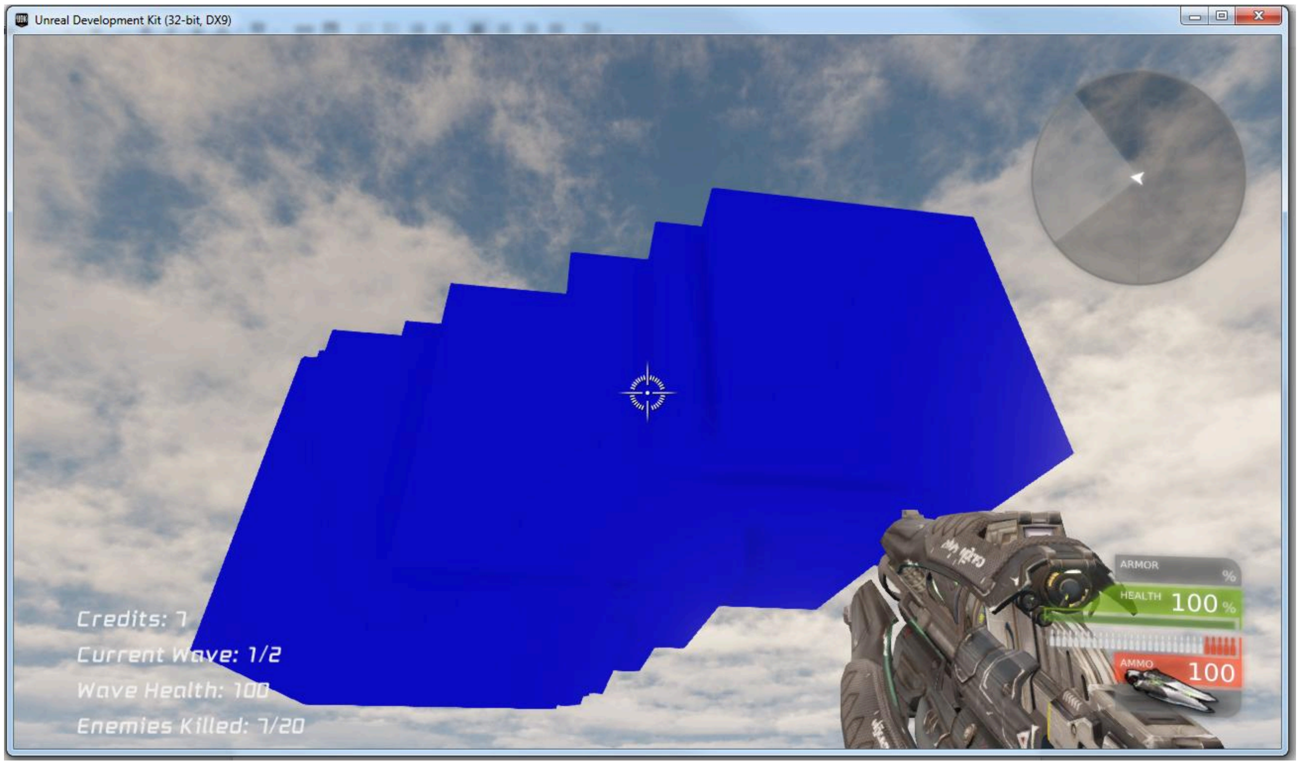
The enemy closest to the player is as low as the enemy will get. I feel like this is still too low so I'm going to increase the height more.



Three times the height of the player seems to work nicely :) I just realised though that i need to change the range on the air enemy's attack range to compensate for the added height. This should just be changing a variable in the defaultproperties. I just doubled the default value for the attack range.

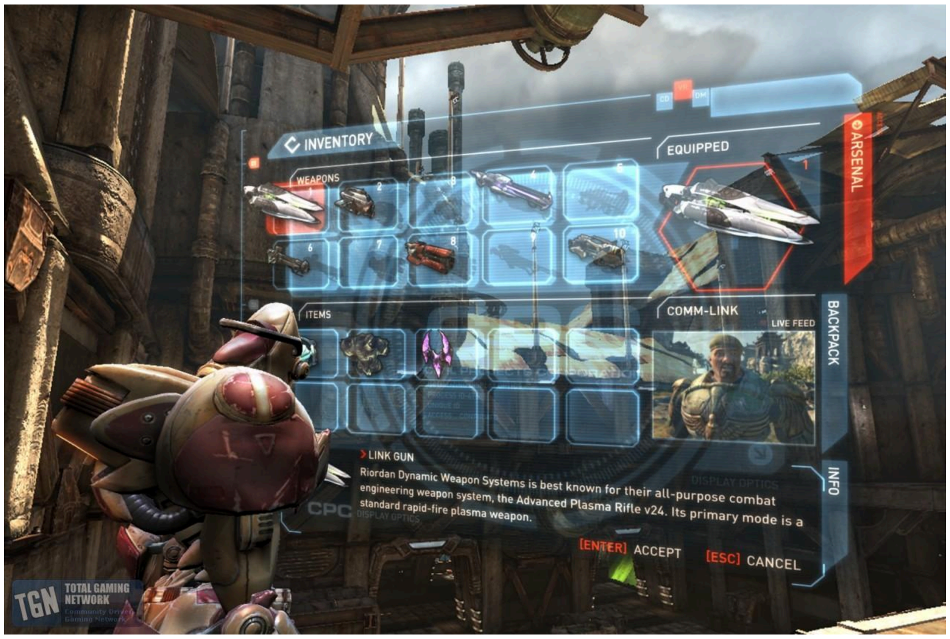


```
defaultproperties
{
	AttackDistance=192.0
	SeekingMat=Material'EditorMaterials.WidgetMaterial_X'
	AttackingMat=Material'EditorMaterials.WidgetMaterial_Z'

	Begin Object Name=EnemyMesh
		Scale3D=(X=0.25,Y=0.25,Z=0.25)
	End Object
}
```



That somewhat sort of worked :L May need to fiddle with it a but yeah

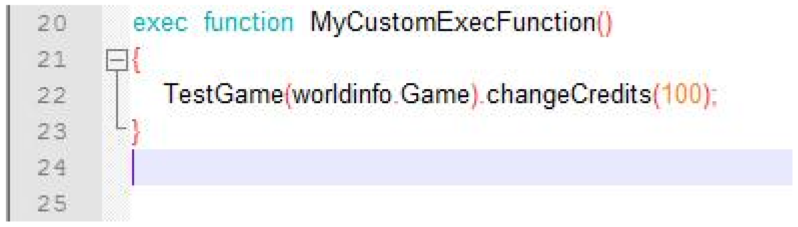
04/02/2012
Today I'm not feeling enemies, I'm going to work on the over shoulder view for the in game menu. What I'm aiming for is something like this but with out the actual menu.



Riordan Dynamic Weapon Systems is best known for their all-purpose combat engineering weapon system, the Advanced Plasma Rifle v24. Its primary mode is a standard rapid-fire plasma weapon.

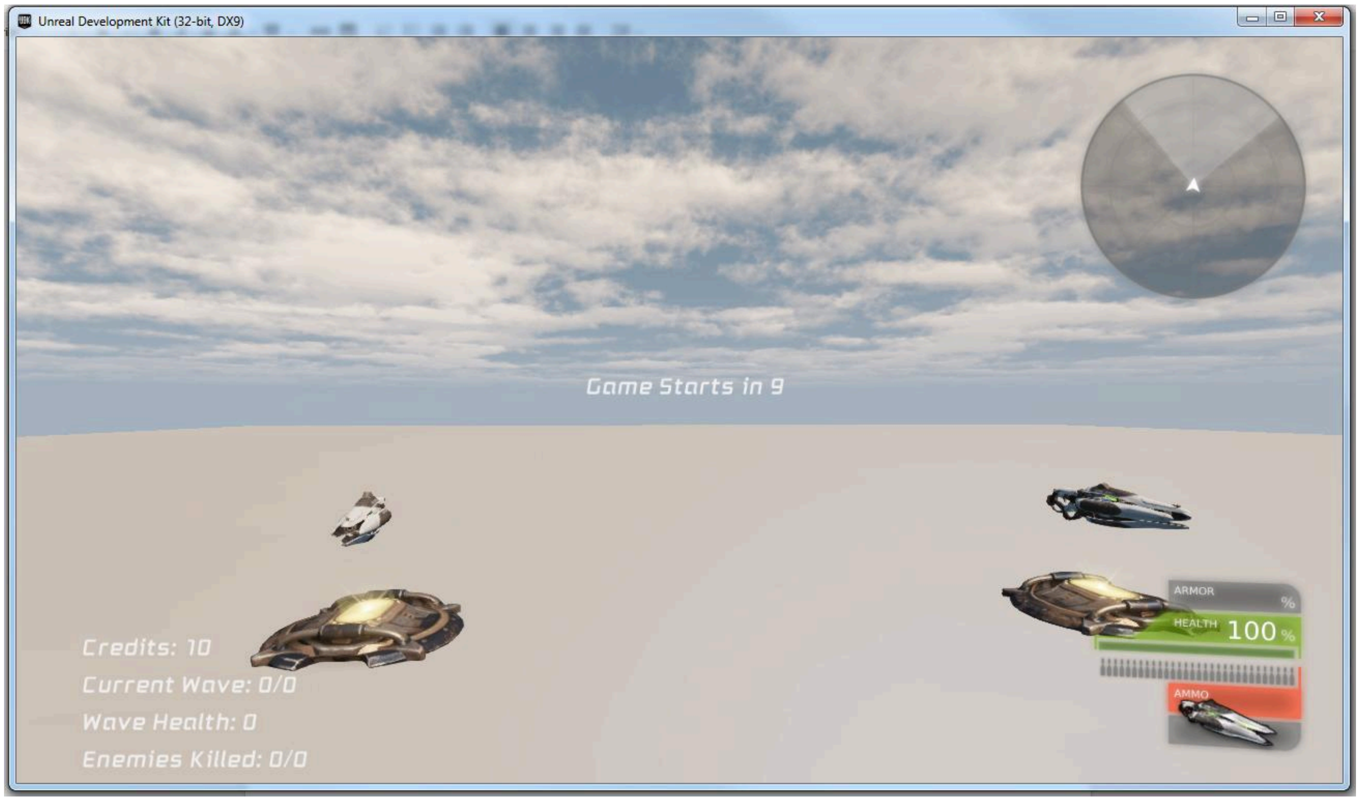
This shouldn't be too difficult, I've covered camera shifting in my UDK book. The one thing i am unsure about is keys. I don't have a clue how to add my own functions in when you press a key so it's time for research! :P Perfect, 5 minutes and I've got a UDK help page (http://udn.epicgames.com/Three/KeyBinds.html) now to read through it.

Right found a lovely bit of code under "Key binding to Unrealscript functions". It shows you how to attach a function to a key. Using this i have changed the function and made it so the you press K you get given 100 credits.

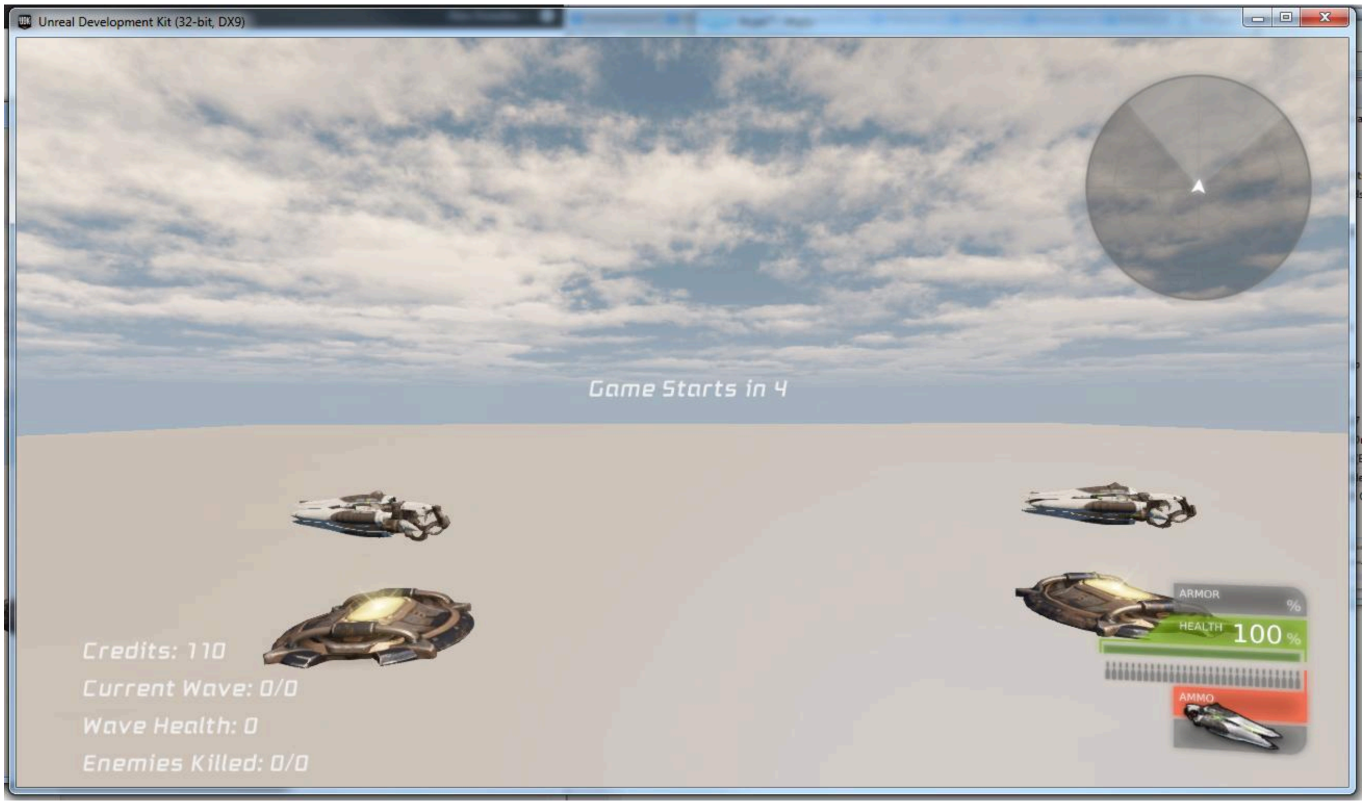


```
exec function MyCustomExecFunction()
{
    TestGame(worldinfo.Game).changeCredits(100);
}
```

[White Space]



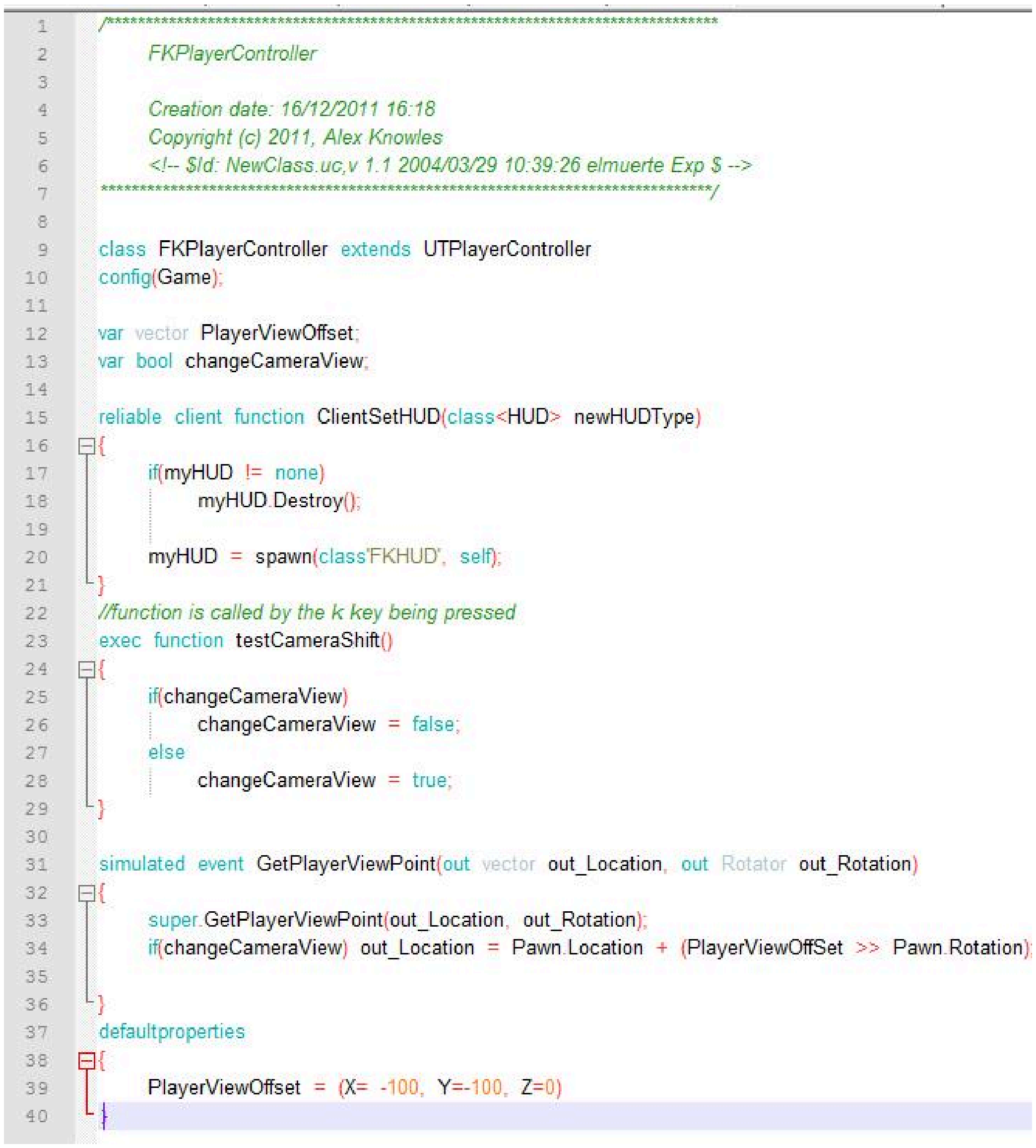
Trust me I pressed K and it was all cool :P



Now to make this move the camera. I know one thing I have to do which is make the player mesh visible to the player when the camera moves away or else we'll be left with this random floating gun.

Okay lets start with renaming the function that's called because MyCustomExecFunction is not very descriptive.I'll just call it testCameraShift. Now just to put in a few lines that change the position of the camera and makes the player mesh visible.

Right this was a bit more difficult than I thought :L I added an if statement so the player can change back and forth the view. This then affect the actual function that handles the player view. Annoyingly I think to make the player mesh visible i need to do that in the pawn it's self.



```
/********************************************************************************
	FKPlayerController

	Creation date: 16/12/2011 16:18
	Copyright (c) 2011, Alex Knowles
	<!-- $Id: NewClass.uc,v 1.1 2004/03/29 10:39:26 elmuerte Exp $ -->
********************************************************************************/

class FKPlayerController extends UTPlayerController
config(Game);

var vector PlayerViewOffset;
var bool changeCameraView;

reliable client function ClientSetHUD(class<HUD> newHUDType)
{
	if(myHUD != none)
		myHUD.Destroy();

	myHUD = spawn(class'FKHUD', self);
}
//function is called by the k key being pressed
exec function testCameraShift()
{
	if(changeCameraView)
		changeCameraView = false;
	else
		changeCameraView = true;
}

simulated event GetPlayerViewPoint(out vector out_Location, out Rotator out_Rotation)
{
	super.GetPlayerViewPoint(out_Location, out_Rotation);
	if(changeCameraView) out_Location = Pawn.Location + (PlayerViewOffSet >> Pawn.Rotation);

}
defaultproperties
{
	PlayerViewOffset = (X= -100, Y=-100, Z=0)
}
```

Thank god i have this UDK book! Following one of the tutorials again I have got the mesh visibility to change. Now just to play with the cameras positioning.

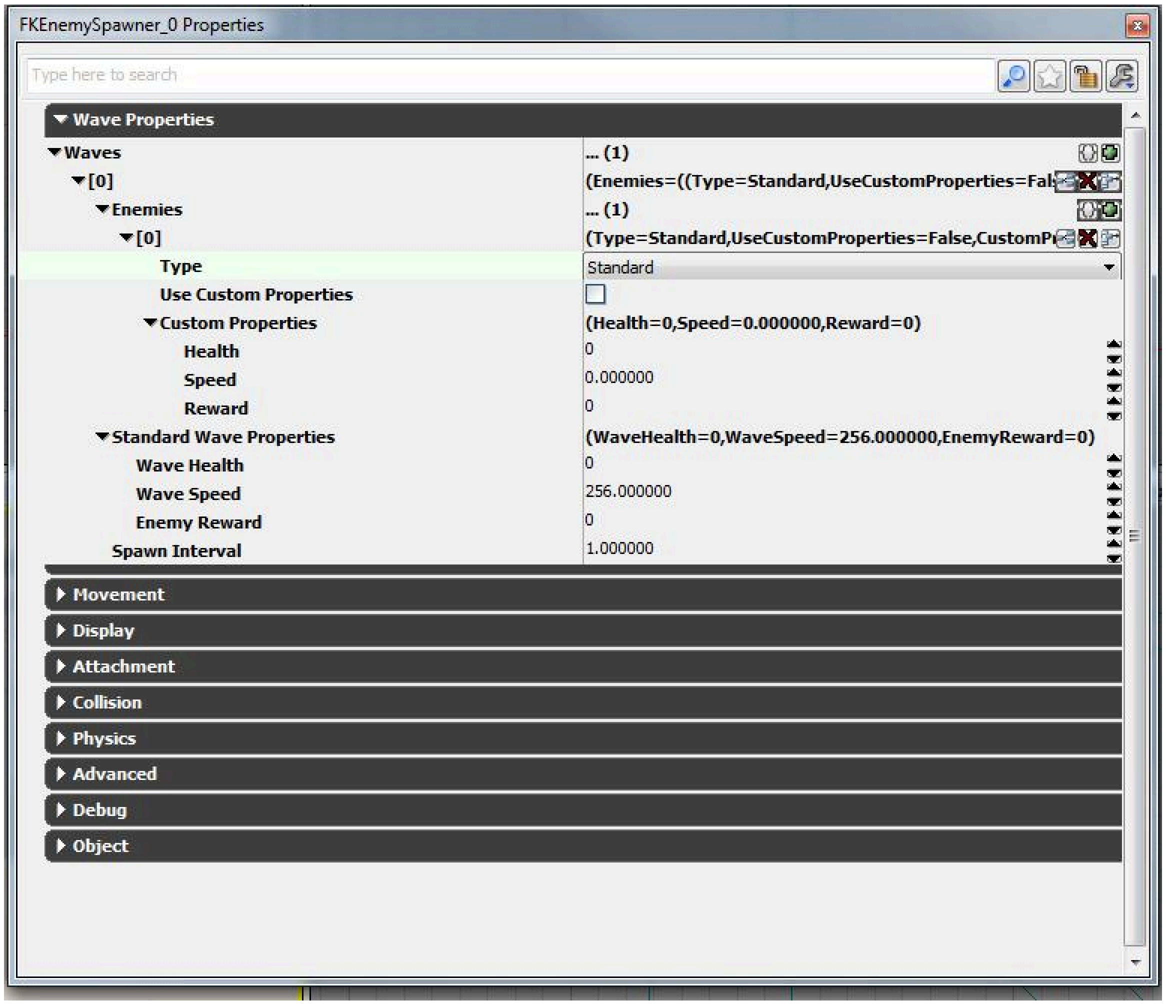
05/02/2012
I’m some what annoyed that I could finish what i was doing yesterday when i had to go to my dads house. I did however make the program which can give the rest of the team my code to play with :P

Anyway back to my over the shoulder menu.

06/02/2012
Not sure what happened yesterday but hey, got nothing done :/ Moving on.

Today I’m going back to enemies. To be more precise, allowing a level designer to make waves without touching code. It’ll be some what like before; being able to choose numbers of enemies, their health, speed, reward and the interval between them, but this time you’ll be able to choose which type to spawn and when to spawn them. So, for example, you may want two air enemies then a tank enemy then 5 standard ones. I want the code to be able to do this and not just like, you can have a wave of just one type because that’s boring. I’m thinking this will be done by using multiple arrays and structures. So you can have basic traits that all the enemies use or custom ones for each enemy. With the basic ones they’re get pushed through an algorithm which makes the right stats for each enemy type. Example, basic health entered is 100, say for a standard enemy that would be something like 50, for a air enemy it would be 100 and for a tank it would be 200. Anyway enough typing, time to make shit :D

First I’m going to work on the layout.

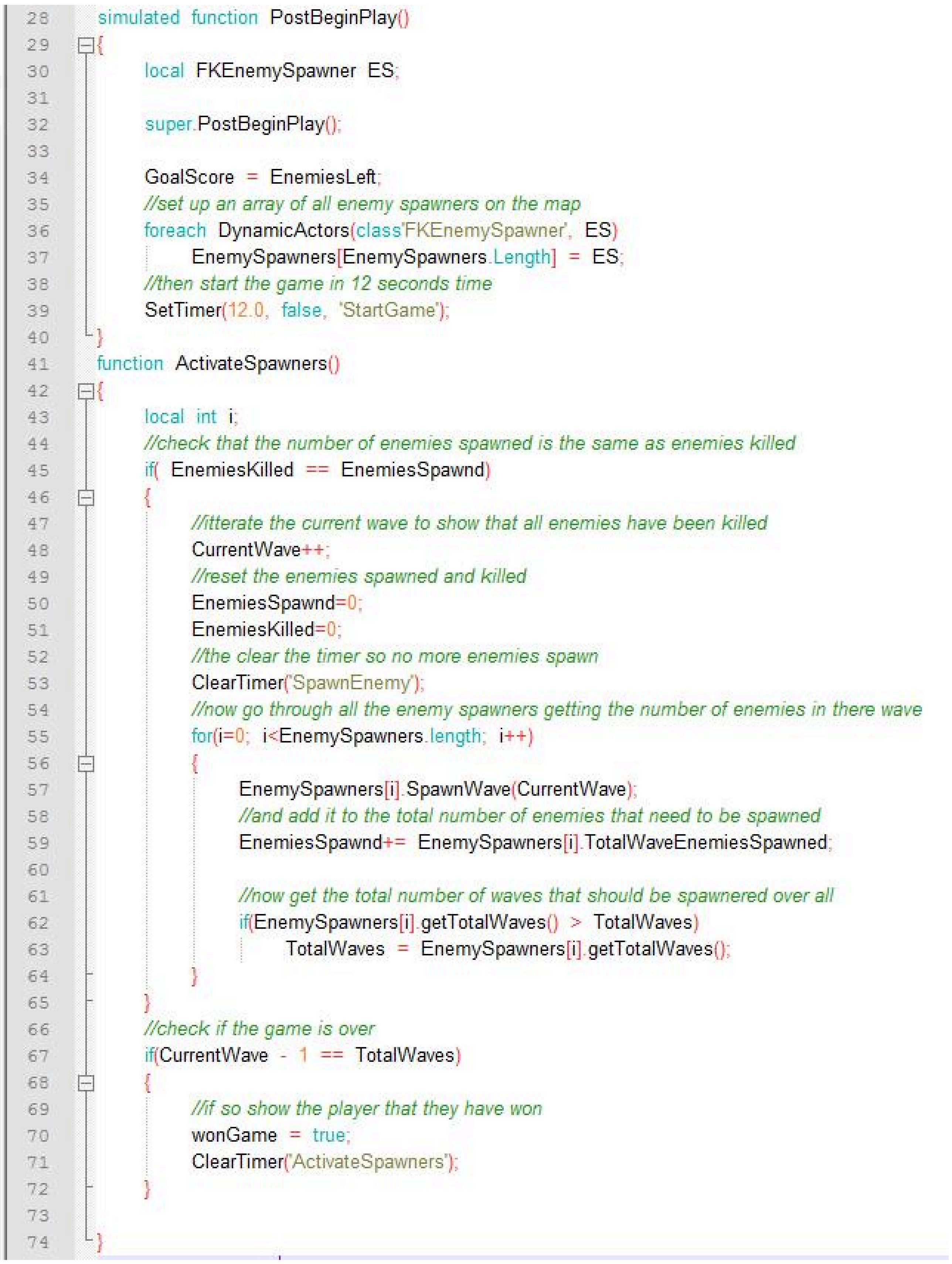


```
FKEnemySpawner_0 Properties
Wave Properties
  Waves                              ... (1)
    [0]                              (Enemies=((Type=Standard,UseCustomProperties=Fal
      Enemies                        ... (1)
        [0]                          (Type=Standard,UseCustomProperties=False,CustomP
            Type                     Standard
            Use Custom Properties    ☐
          Custom Properties          (Health=0,Speed=0.000000,Reward=0)
              Health                 0
              Speed                  0.000000
              Reward                 0
      Standard Wave Properties       (WaveHealth=0,WaveSpeed=256.000000,EnemyReward=0)
          Wave Health                0
          Wave Speed                 256.000000
          Enemy Reward               0
      Spawn Interval                 1.000000
Movement
Display
Attachment
Collision
Physics
Advanced
Debug
Object
```

08/02/2012
Now that I've sorted out the layout time to make it all work :D I should probably right some pseudo code first but hey like to jump in the deep end.

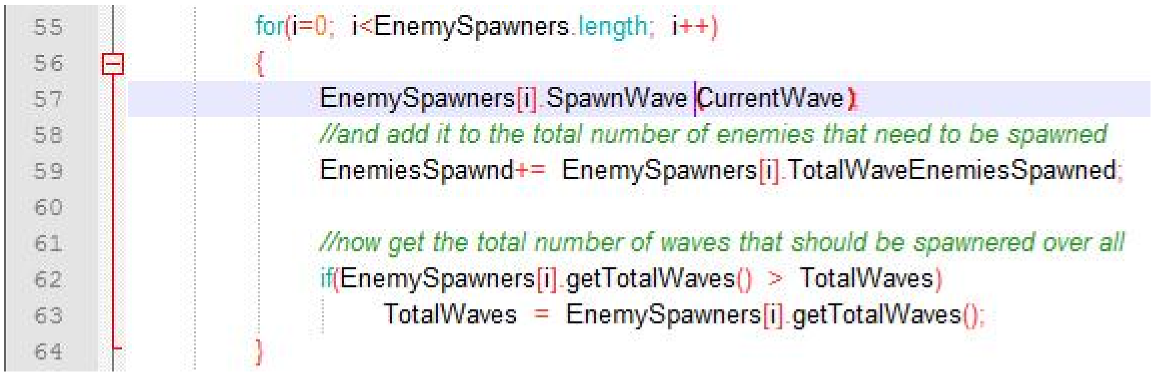
I don't think the code for the main game needs to changed much, just the way the total number of enemies is found. Actually reading through it again i should be fine because everything is just calls to functions in the spawn itself.

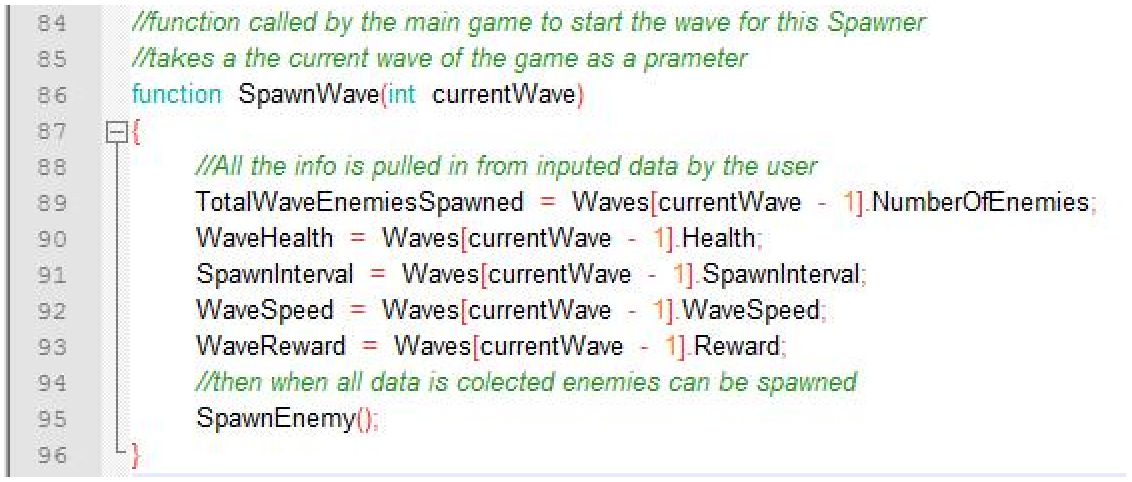
(There is more to the code but it's not really relative to what I'm talking about)



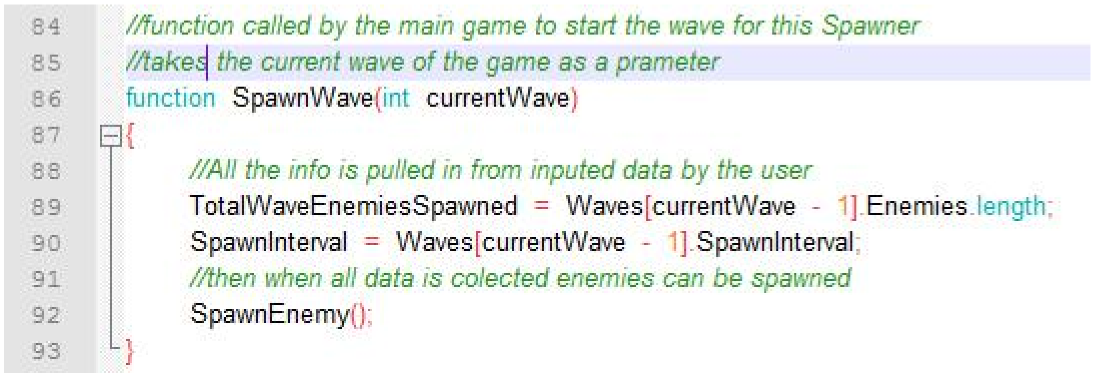
```
simulated function PostBeginPlay()
{
    local FKEnemySpawner ES;

    super.PostBeginPlay();

    GoalScore = EnemiesLeft;
    //set up an array of all enemy spawners on the map
    foreach DynamicActors(class'FKEnemySpawner', ES)
        EnemySpawners[EnemySpawners.Length] = ES;
    //then start the game in 12 seconds time
    SetTimer(12.0, false, 'StartGame');
}
function ActivateSpawners()
{
    local int i;
    //check that the number of enemies spawned is the same as enemies killed
    if( EnemiesKilled == EnemiesSpawnd)
    {
        //itterate the current wave to show that all enemies have been killed
        CurrentWave++;
        //reset the enemies spawned and killed
        EnemiesSpawnd=0;
        EnemiesKilled=0;
        //the clear the timer so no more enemies spawn
        ClearTimer('SpawnEnemy');
        //now go through all the enemy spawners getting the number of enemies in there wave
        for(i=0; i<EnemySpawners.length; i++)
        {
            EnemySpawners[i].SpawnWave(CurrentWave);
            //and add it to the total number of enemies that need to be spawned
            EnemiesSpawnd+= EnemySpawners[i].TotalWaveEnemiesSpawned;

            //now get the total number of waves that should be spawnered over all
            if(EnemySpawners[i].getTotalWaves() > TotalWaves)
                TotalWaves = EnemySpawners[i].getTotalWaves();
        }
    }
    //check if the game is over
    if(CurrentWave - 1 == TotalWaves)
    {
        //if so show the player that they have won
        wonGame = true;
        ClearTimer('ActivateSpawners');
    }

}
```

This mean that I may need to totally rewrite the EnemySpawner class -.- May as well go through it logically, starting with the first function that is called by the main game. This is SpawnWave(CurrentWave)

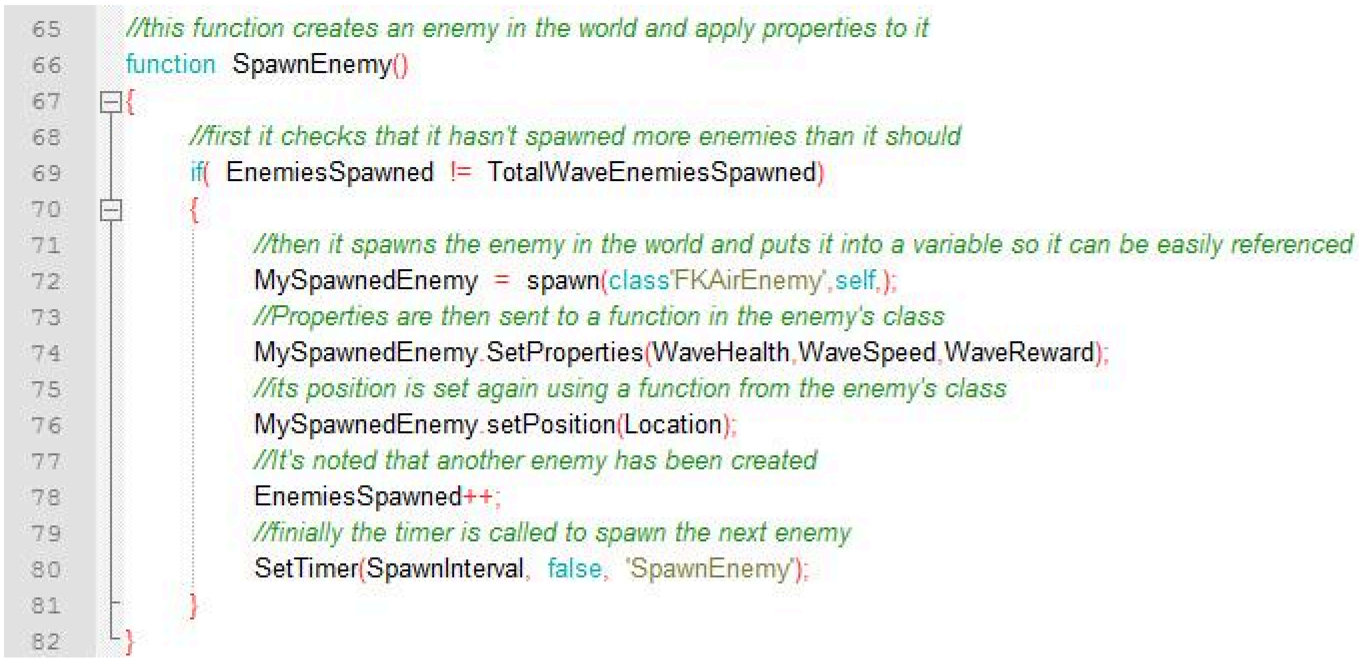

```
            for(i=0; i<EnemySpawners.length; i++)
            {
                EnemySpawners[i].SpawnWave(CurrentWave);
                //and add it to the total number of enemies that need to be spawned
                EnemiesSpawnd+= EnemySpawners[i].TotalWaveEnemiesSpawned;

                //now get the total number of waves that should be spawnered over all
                if(EnemySpawners[i].getTotalWaves() > TotalWaves)
                    TotalWaves = EnemySpawners[i].getTotalWaves();
            }
```


```
    //function called by the main game to start the wave for this Spawner
    //takes a the current wave of the game as a prameter
    function SpawnWave(int currentWave)
    {
        //All the info is pulled in from inputed data by the user
        TotalWaveEnemiesSpawned = Waves[currentWave - 1].NumberOfEnemies;
        WaveHealth = Waves[currentWave - 1].Health;
        SpawnInterval = Waves[currentWave - 1].SpawnInterval;
        WaveSpeed = Waves[currentWave - 1].WaveSpeed;
        WaveReward = Waves[currentWave - 1].Reward;
        //then when all data is colected enemies can be spawned
        SpawnEnemy();
    }
```

All I need to do is change were it’s pulling the inputted data from, I think. Okay I had to remove health, wave speed and reward because they’re just too general and should be check when the enemy is being spawned so it can see if it needs to load in custom variables or not.


```
    //function called by the main game to start the wave for this Spawner
    //takes the current wave of the game as a prameter
    function SpawnWave(int currentWave)
    {
        //All the info is pulled in from inputed data by the user
        TotalWaveEnemiesSpawned = Waves[currentWave - 1].Enemies.length;
        SpawnInterval = Waves[currentWave - 1].SpawnInterval;
        //then when all data is colected enemies can be spawned
        SpawnEnemy();
    }
```

SpawnEnemy, now this should take a few more lines than currently there.



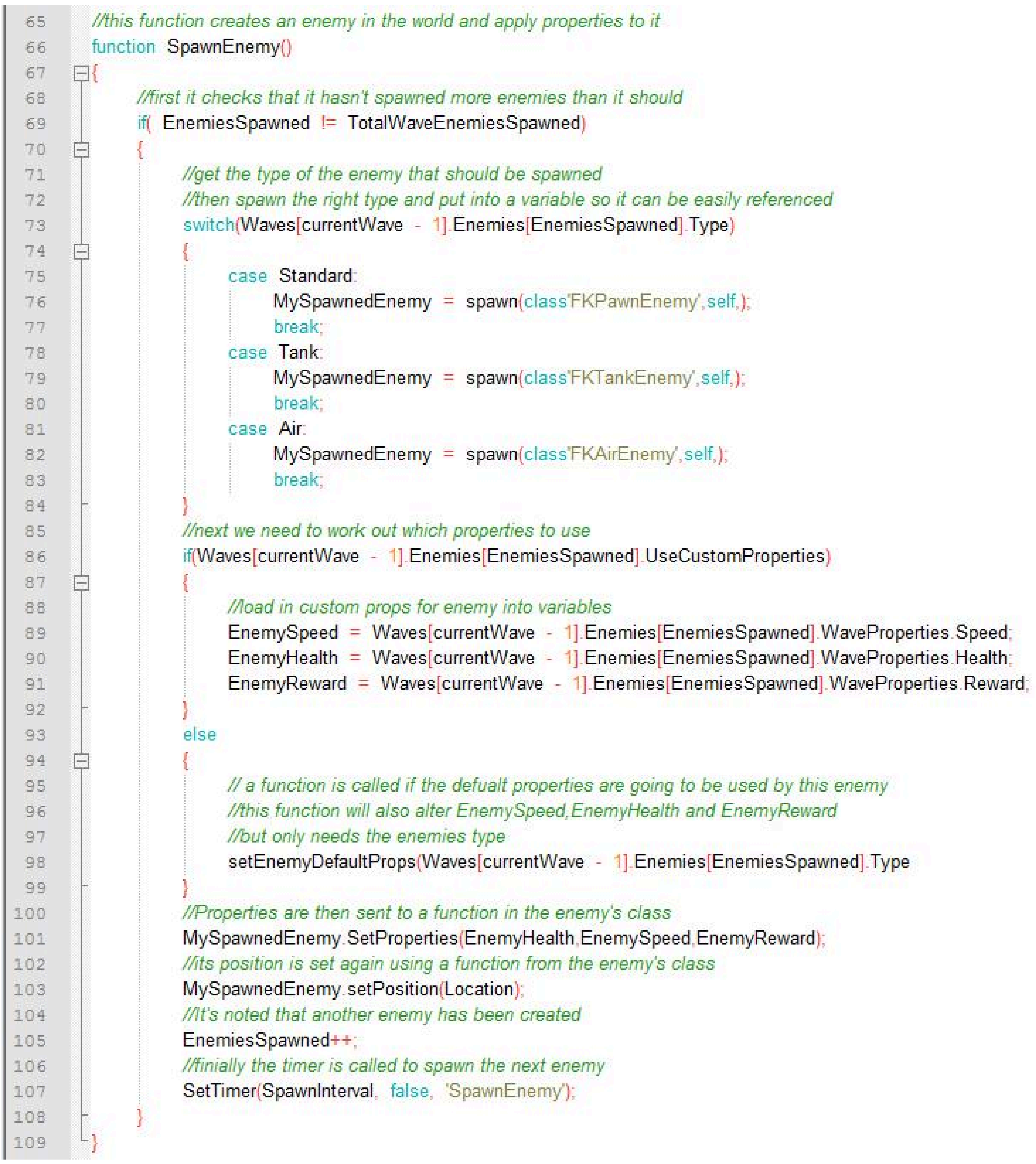
```
//this function creates an enemy in the world and apply properties to it
function SpawnEnemy()
{
	//first it checks that it hasn't spawned more enemies than it should
	if( EnemiesSpawned != TotalWaveEnemiesSpawned)
	{
		//then it spawns the enemy in the world and puts it into a variable so it can be easily referenced
		MySpawnedEnemy = spawn(class'FKAirEnemy',self,);
		//Properties are then sent to a function in the enemy's class
		MySpawnedEnemy.SetProperties(WaveHealth,WaveSpeed,WaveReward);
		//its position is set again using a function from the enemy's class
		MySpawnedEnemy.setPosition(Location);
		//It's noted that another enemy has been created
		EnemiesSpawned++;
		//finially the timer is called to spawn the next enemy
		SetTimer(SpawnInterval, false, 'SpawnEnemy');
	}
}
```

Now with this it depends on all data pulled in before it was called and only one enemy type can be spawned. I've got an idea of how I'm going to do it but I can't be bother to write it down I'm just going to program it because I'm one lazy motherfucker.

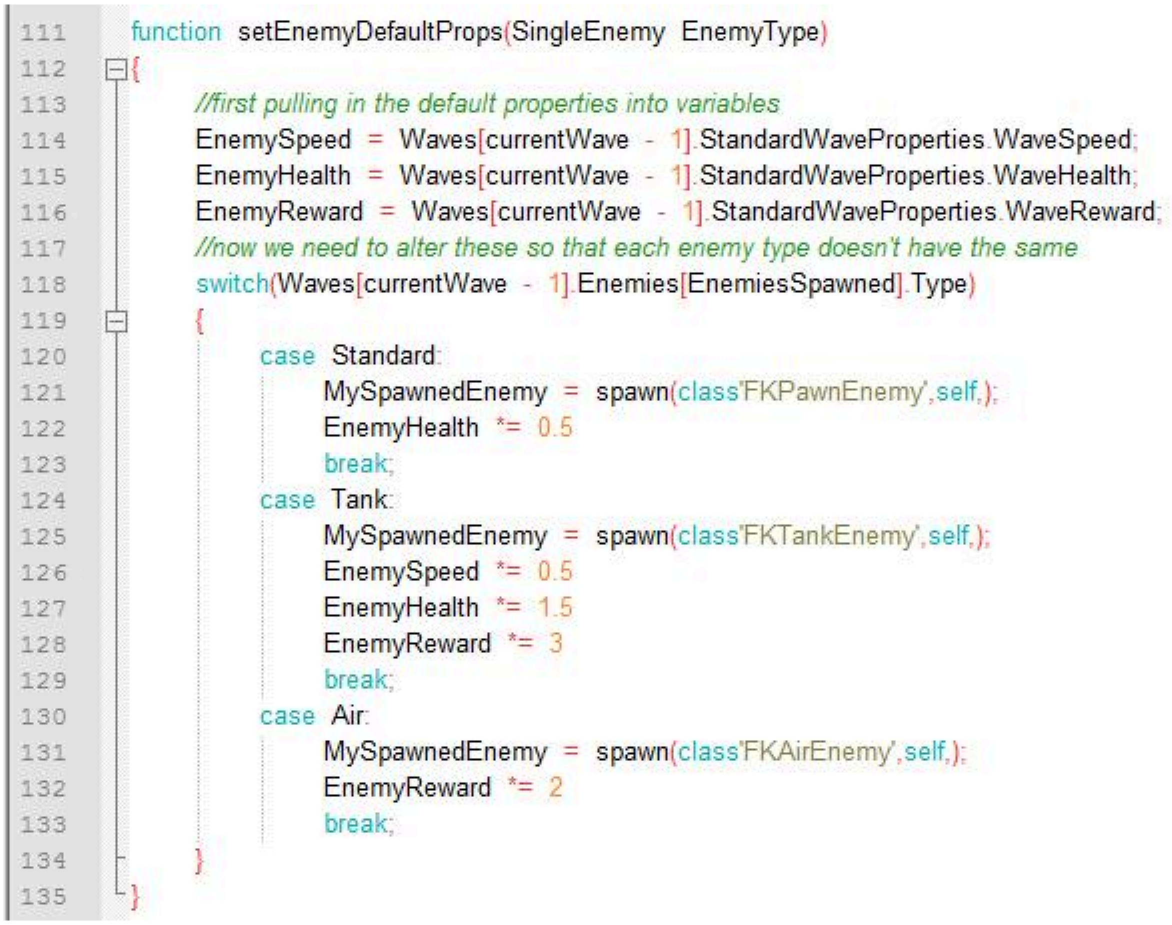
09/02/2012
I seem to have a problem I either right comments in code and don't document or don't comment and explain it all here :/

Anyway I got a lot of the code done yesterday and now I've moved on to the algorithm which alters the default properties depending on each enemy type.

Here's the spawn enemy function anyway. Haven't done any debugging yet



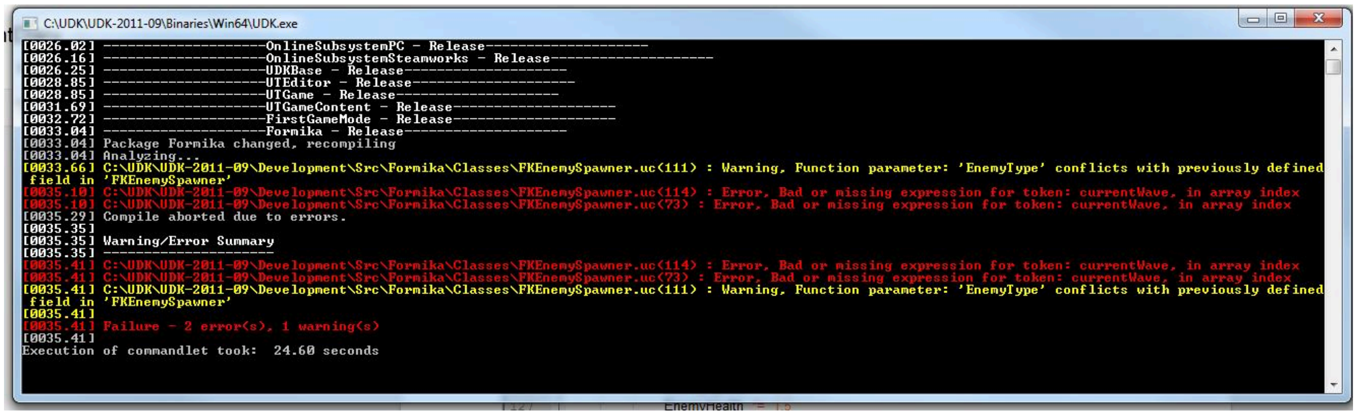
```
//this function creates an enemy in the world and apply properties to it
function SpawnEnemy()
{
    //first it checks that it hasn't spawned more enemies than it should
    if( EnemiesSpawned != TotalWaveEnemiesSpawned)
    {
        //get the type of the enemy that should be spawned
        //then spawn the right type and put into a variable so it can be easily referenced
        switch(Waves[currentWave - 1].Enemies[EnemiesSpawned].Type)
        {
            case Standard:
                MySpawnedEnemy = spawn(class'FKPawnEnemy',self,);
                break;
            case Tank:
                MySpawnedEnemy = spawn(class'FKTankEnemy',self,);
                break;
            case Air:
                MySpawnedEnemy = spawn(class'FKAirEnemy',self,);
                break;
        }
        //next we need to work out which properties to use
        if(Waves[currentWave - 1].Enemies[EnemiesSpawned].UseCustomProperties)
        {
            //load in custom props for enemy into variables
            EnemySpeed = Waves[currentWave - 1].Enemies[EnemiesSpawned].WaveProperties.Speed;
            EnemyHealth = Waves[currentWave - 1].Enemies[EnemiesSpawned].WaveProperties.Health;
            EnemyReward = Waves[currentWave - 1].Enemies[EnemiesSpawned].WaveProperties.Reward;
        }
        else
        {
            // a function is called if the defualt properties are going to be used by this enemy
            //this function will also alter EnemySpeed,EnemyHealth and EnemyReward
            //but only needs the enemies type
            setEnemyDefaultProps(Waves[currentWave - 1].Enemies[EnemiesSpawned].Type
        }
        //Properties are then sent to a function in the enemy's class
        MySpawnedEnemy.SetProperties(EnemyHealth,EnemySpeed,EnemyReward);
        //its position is set again using a function from the enemy's class
        MySpawnedEnemy.setPosition(Location);
        //It's noted that another enemy has been created
        EnemiesSpawned++;
        //finially the timer is called to spawn the next enemy
        SetTimer(SpawnInterval, false, 'SpawnEnemy');
    }
}
```



```
111  function setEnemyDefaultProps(SingleEnemy EnemyType)
112  {
113      //first pulling in the default properties into variables
114      EnemySpeed = Waves[currentWave - 1].StandardWaveProperties.WaveSpeed;
115      EnemyHealth = Waves[currentWave - 1].StandardWaveProperties.WaveHealth;
116      EnemyReward = Waves[currentWave - 1].StandardWaveProperties.WaveReward;
117      //now we need to alter these so that each enemy type doesn't have the same
118      switch(Waves[currentWave - 1].Enemies[EnemiesSpawned].Type)
119      {
120          case Standard:
121              MySpawnedEnemy = spawn(class'FKPawnEnemy',self,);
122              EnemyHealth *= 0.5
123              break;
124          case Tank:
125              MySpawnedEnemy = spawn(class'FKTankEnemy',self,);
126              EnemySpeed *= 0.5
127              EnemyHealth *= 1.5
128              EnemyReward *= 3
129              break;
130          case Air:
131              MySpawnedEnemy = spawn(class'FKAirEnemy',self,);
132              EnemyReward *= 2
133              break;
134      }
135  }
```

Okay I've written the setEnemyDefaultProps function now I just need it so the enemy itself get the properties. I've made a call to SetProperties which is part of FKEnemy class so i should go change that to cope with the new variable I want to send through.

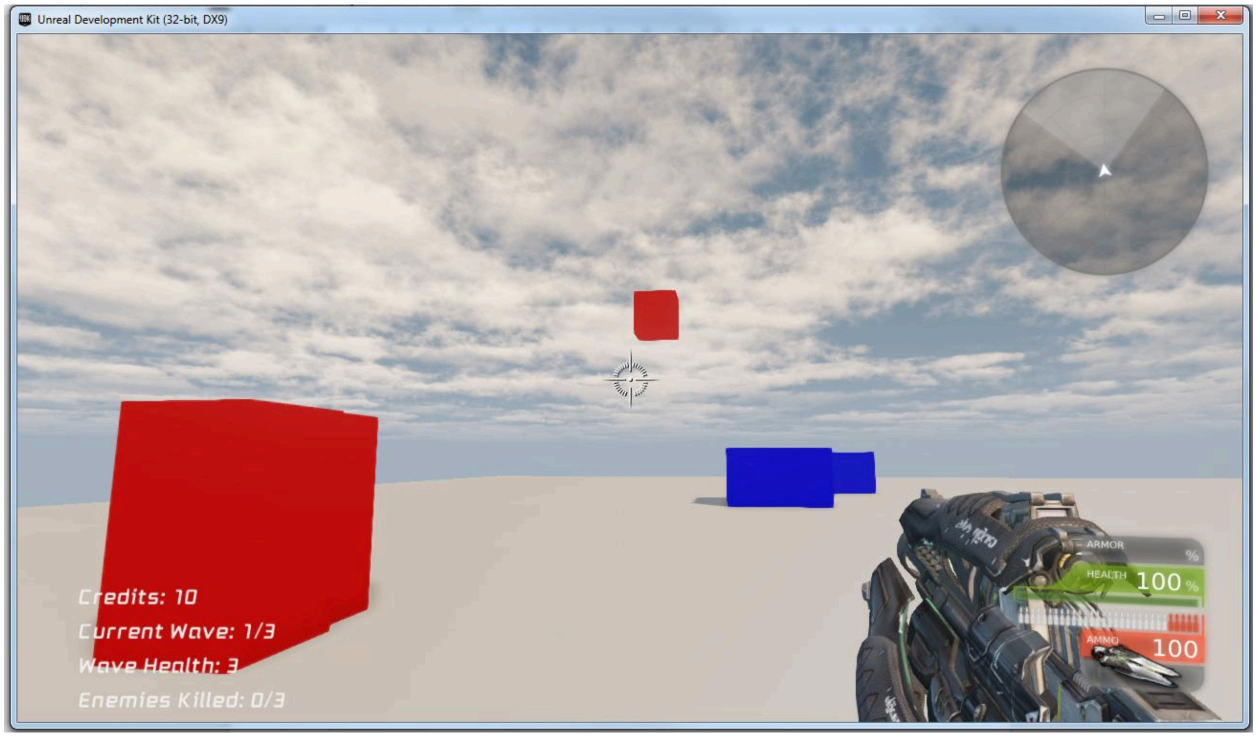
Okay I was wrong the function already works with the code. Time to compile!



```
C:\UDK\UDK-2011-09\Binaries\Win64\UDK.exe
[0026.02] --------------------OnlineSubsystemPC - Release--------------------
[0026.16] --------------------OnlineSubsystemSteamworks - Release--------------------
[0026.25] --------------------UDKBase - Release--------------------
[0028.85] --------------------UTEditor - Release--------------------
[0028.85] --------------------UTGame - Release--------------------
[0031.69] --------------------UTGameContent - Release--------------------
[0032.72] --------------------FirstGameMode - Release--------------------
[0033.04] --------------------Formika - Release--------------------
[0033.04] Package Formika changed, recompiling
[0033.04] Analyzing...
[0033.66] C:\UDK\UDK-2011-09\Development\Src\Formika\Classes\FKEnemySpawner.uc(111) : Warning, Function parameter: 'EnemyType' conflicts with previously defined
 field in 'FKEnemySpawner'
[0035.10] C:\UDK\UDK-2011-09\Development\Src\Formika\Classes\FKEnemySpawner.uc(114) : Error, Bad or missing expression for token: currentWave, in array index
[0035.10] C:\UDK\UDK-2011-09\Development\Src\Formika\Classes\FKEnemySpawner.uc(73) : Error, Bad or missing expression for token: currentWave, in array index
[0035.29] Compile aborted due to errors.
[0035.35]
[0035.35] Warning/Error Summary
[0035.35] ---------------------
[0035.41] C:\UDK\UDK-2011-09\Development\Src\Formika\Classes\FKEnemySpawner.uc(114) : Error, Bad or missing expression for token: currentWave, in array index
[0035.41] C:\UDK\UDK-2011-09\Development\Src\Formika\Classes\FKEnemySpawner.uc(73) : Error, Bad or missing expression for token: currentWave, in array index
[0035.41] C:\UDK\UDK-2011-09\Development\Src\Formika\Classes\FKEnemySpawner.uc(111) : Warning, Function parameter: 'EnemyType' conflicts with previously defined
 field in 'FKEnemySpawner'
[0035.41]
[0035.41] Failure - 2 error(s), 1 warning(s)
[0035.41]
Execution of commandlet took:  24.60 seconds
```

Wow, I made some silly little syntax errors -.-

27/02/2012
Hmm I really haven't done anything in awhile :/ So last I was working on the new spawning system and currently this only spawns one wave and then doesn't work. I thinking there's a problem when it tries to spawn in the next wave.

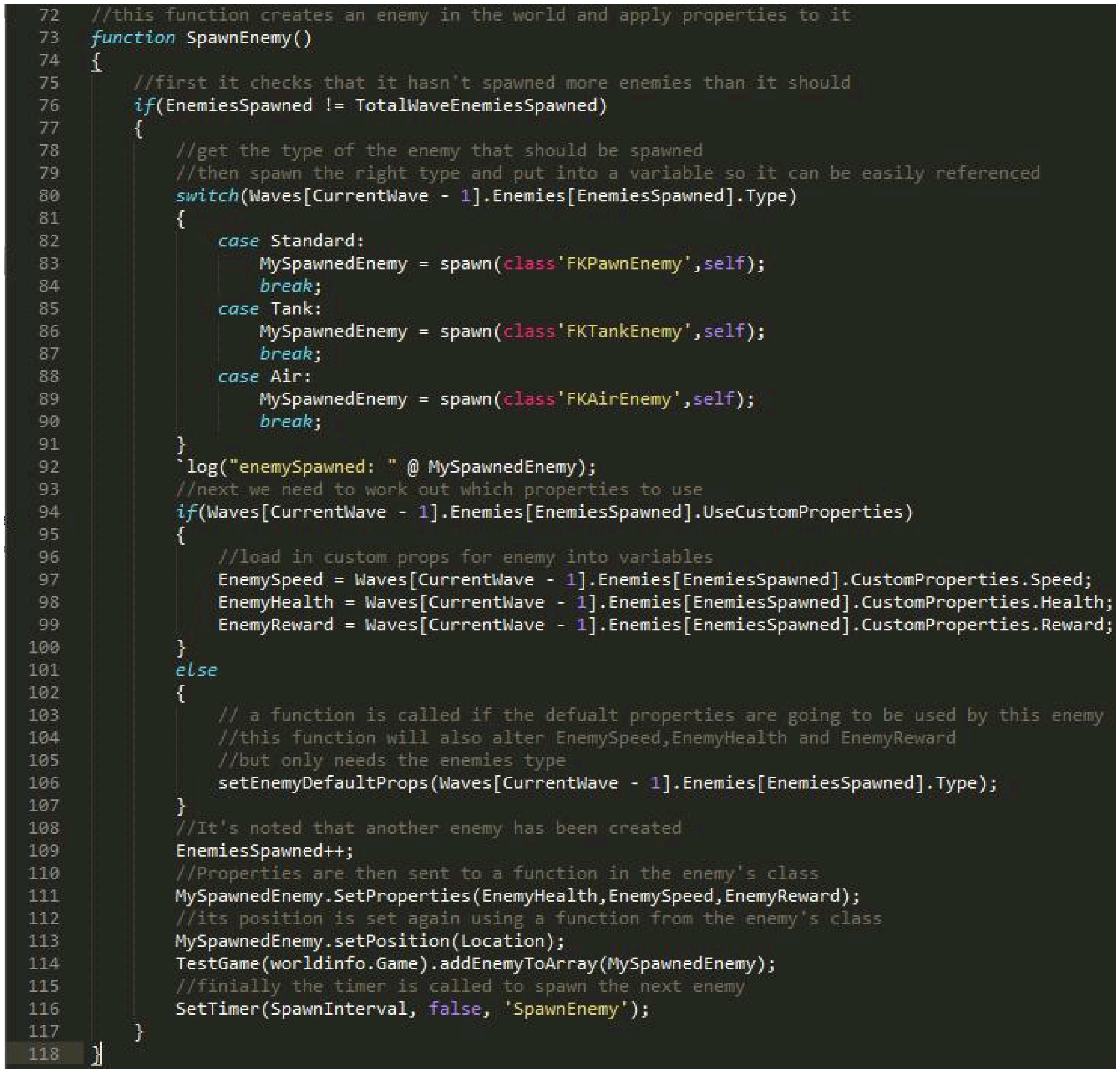


Credits: 10
Current Wave: 1/3
Wave Health: 3
Enemies Killed: 0/3

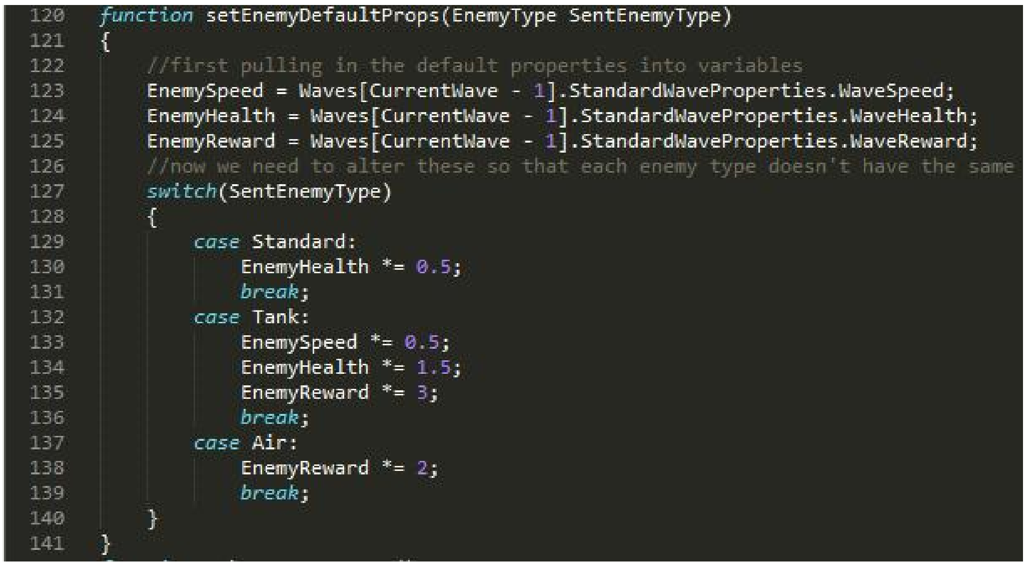
I've set the game up to spawn 1 of each enemy type on the first round. For some reason its spawned 2 standard, 2 tanks and 1 air enemy.

I found that I set the code to spawn two enemy for each type -.- and so yeah I've got that sorted and I'm planning to put a video out soon.

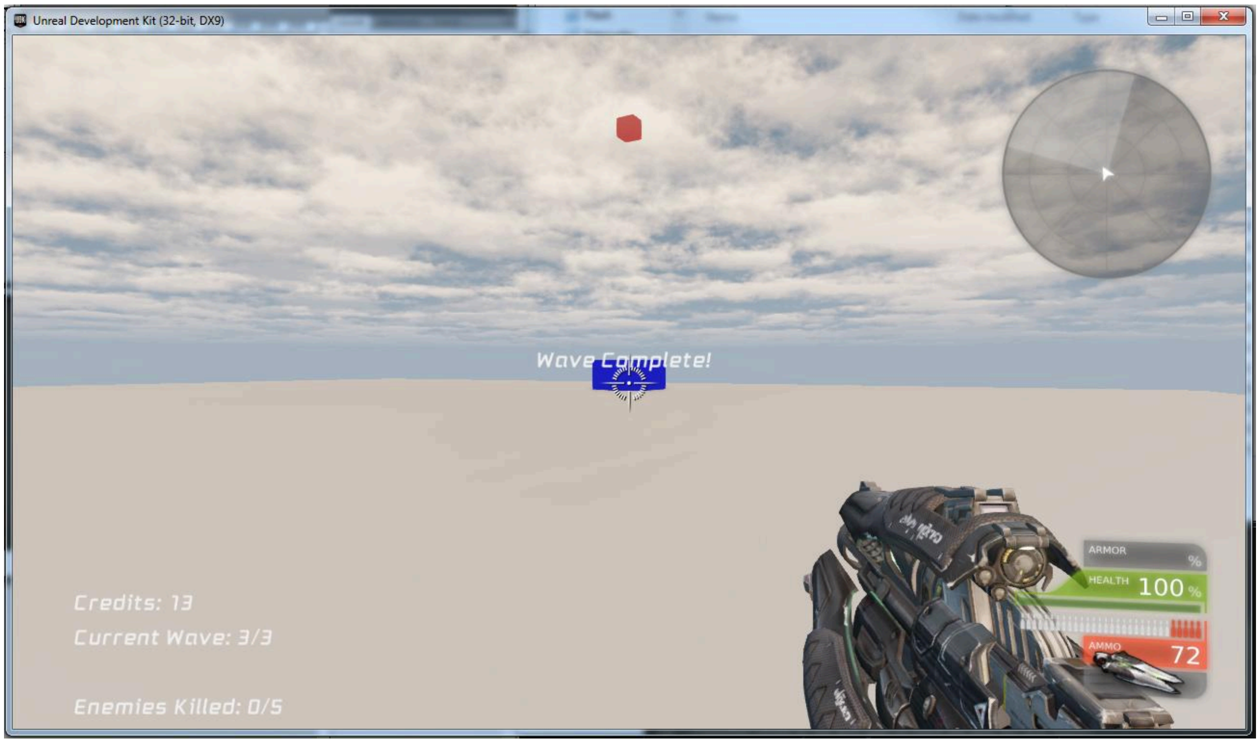
05/03/2012
I did do some other stuff to the SpawnEnemy & setEnemyDefaultProperties function but I can't remember what they are so I'll just re-post the new code:



```
//this function creates an enemy in the world and apply properties to it
function SpawnEnemy()
{
    //first it checks that it hasn't spawned more enemies than it should
    if(EnemiesSpawned != TotalWaveEnemiesSpawned)
    {
        //get the type of the enemy that should be spawned
        //then spawn the right type and put into a variable so it can be easily referenced
        switch(Waves[CurrentWave - 1].Enemies[EnemiesSpawned].Type)
        {
            case Standard:
                MySpawnedEnemy = spawn(class'FKPawnEnemy',self);
                break;
            case Tank:
                MySpawnedEnemy = spawn(class'FKTankEnemy',self);
                break;
            case Air:
                MySpawnedEnemy = spawn(class'FKAirEnemy',self);
                break;
        }
        `log("enemySpawned: " @ MySpawnedEnemy);
        //next we need to work out which properties to use
        if(Waves[CurrentWave - 1].Enemies[EnemiesSpawned].UseCustomProperties)
        {
            //load in custom props for enemy into variables
            EnemySpeed = Waves[CurrentWave - 1].Enemies[EnemiesSpawned].CustomProperties.Speed;
            EnemyHealth = Waves[CurrentWave - 1].Enemies[EnemiesSpawned].CustomProperties.Health;
            EnemyReward = Waves[CurrentWave - 1].Enemies[EnemiesSpawned].CustomProperties.Reward;
        }
        else
        {
            // a function is called if the defualt properties are going to be used by this enemy
            //this function will also alter EnemySpeed,EnemyHealth and EnemyReward
            //but only needs the enemies type
            setEnemyDefaultProps(Waves[CurrentWave - 1].Enemies[EnemiesSpawned].Type);
        }
        //It's noted that another enemy has been created
        EnemiesSpawned++;
        //Properties are then sent to a function in the enemy's class
        MySpawnedEnemy.SetProperties(EnemyHealth,EnemySpeed,EnemyReward);
        //its position is set again using a function from the enemy's class
        MySpawnedEnemy.setPosition(Location);
        TestGame(worldinfo.Game).addEnemyToArray(MySpawnedEnemy);
        //finially the timer is called to spawn the next enemy
        SetTimer(SpawnInterval, false, 'SpawnEnemy');
    }
}
```



```
function setEnemyDefaultProps(EnemyType SentEnemyType)
{
    //first pulling in the default properties into variables
    EnemySpeed = Waves[CurrentWave - 1].StandardWaveProperties.WaveSpeed;
    EnemyHealth = Waves[CurrentWave - 1].StandardWaveProperties.WaveHealth;
    EnemyReward = Waves[CurrentWave - 1].StandardWaveProperties.WaveReward;
    //now we need to alter these so that each enemy type doesn't have the same
    switch(SentEnemyType)
    {
        case Standard:
            EnemyHealth *= 0.5;
            break;
        case Tank:
            EnemySpeed *= 0.5;
            EnemyHealth *= 1.5;
            EnemyReward *= 3;
            break;
        case Air:
            EnemyReward *= 2;
            break;
    }
}
```



When I run the code now, the waves spawn correctly with no extras. One problem I do have is that enemies can go through each other.