**CMP4183 Game Engines;**

Our group consisted of myself, Chelsea Cafolla, Jamie Evans and Vlad Dumitriu. The game engine we were tasked with to use was Construct2, the genre of game we planned to make was a rogue-like dungeon crawler with graphics being pixel art to stick with the theme of retro dungeon crawlers. From the beginning, we decided to split the different game into different sections that would be individually worked on. These would then make the up the different tutorials, these included player controls and variables, enemy movements and variables, power-ups and the environment and background. This report will focus on what we as a team could do with our engine and how the other game engines taught compared to the engine. This report will also feature each of my team’s tutorials that will highlight the different features of our game and how we made them possible.

The game engine we used, Construct2, was the free edition with limited functions. The engine allowed for simple yet effective programming for our game, this is because the engine used events to create the game and how it interacts with the different objects in-game. The engine used variables and properties to create multiple game features like the player’s health and the players score. The engine was effective and simple to use meaning there was no steep learning curve, the basic tutorial provided by the developers of the engine shows in-depth steps on how to create different features of a game like creating multiple enemies and player controls. The engine allows for large amounts of customisation of the many features of the game, an example being the canvas size can be altered to create different sized levels. Another feature of the engine is behaviours which gives certain properties to game objects, these including bullet meaning the object will travel in one direction at a designated speed one other behaviour being bound to layout which prevents the objects exiting the layout. These many features allow for precise and effective manipulation of game objects, which were shown in the tutorials. However, this engine does have problems, Construct2 is limited the number of events that was possible on one project, this put a constraint onto what we could do for our game which in turn affected our tutorials as we couldn’t teach everything we wanted.

Tutorials;

Michael Cullen, Power-ups;

How to create modifiers within construct 2;

Step 1;

Find or create images that you find fitting to use as your power-ups. You will need one for a damage modifier and another for a health-pack.

Step 2;

Insert the power-ups onto the layout, you should also add a player with the ability to shoot. Once that is done we can begin to create events for them.

Step 3;

Go to the event sheet. Then click ‘Add event’ after this select the player sprite and the condition ‘is overlapping’. To find this you can also use the search bar. Then select the sprite for the health pack.

Step 4;

Then we want to change the player’s health instance variable. Select the player sprite and add the action of ‘add to’ under the variable section. Then add 1 to player health variable.

Step 5;

When the player goes over the health pack the player’s health will increase by one. However, the health pack is still there. Add another action, select the health pack sprite and then click ‘destroy’.

Step 6;

To increase the damage, add the following events. Select the player sprite and add the condition ‘is overlapping’. For this section, you will need an enemy. Then add another event so that when the player is overlapping the modifier the players bullet sprite when, ‘on collision’ with enemy ‘subtract from enemy health’ 3. This creates the affect that the player is doing more damage.

Chelsea Cafolla, Environment/Background;

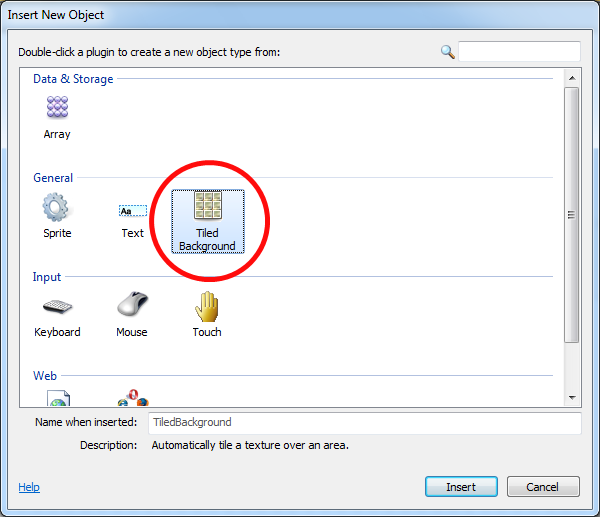
Level Design Tutorial

Tiled Backgrounds

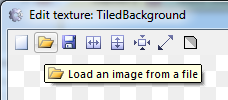
1. Create or download an appropriate image. A tile which resembles a top down view will work best in this instance. Remember, power-of-two sized images (16x16, 32x32, 64x64, 128x128 etc.) will achieve the most effective outcome. Non-power-of-two images will still work, however, Construct2 will resize them to P.O.T by default so they may stretch.



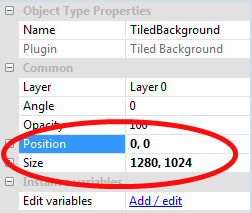
1. In Construct 2, Insert a new object into the layout. In the new window, select “Tiled Background”.



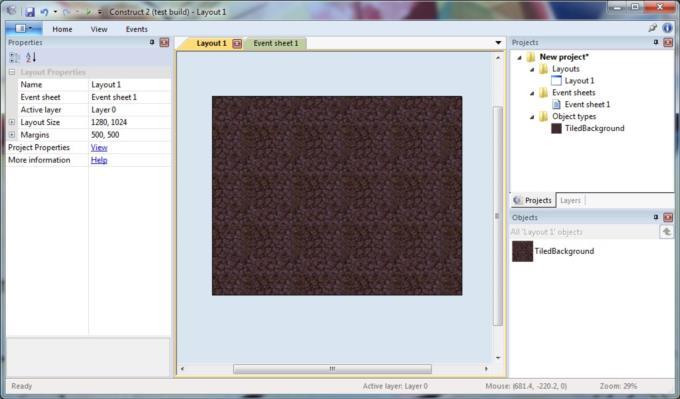
1. A crosshair will now appear over the cursor. Use this to indicate where to place the object. For best results, select somewhere in the middle of the layout. A new window will open; this will be the texture editor. To choose the tile in which was previously created/saved, click on the folder icon and locate and select the image from its location in the file explorer.



1. Close the window and your tiled background object should appear on the layout. This may not be the full size of the layout just yet. To achieve this, it can either be rescaled manually by dragging it, or to be more accurate, go to the Properties Bar (located on the left) and find the size and position section.



1. To ensure that the new background completely fills the layout, zoom in and out by holding ctrl and the mouse wheel then scrolling up or down. Alternatively, this can be achieved by selecting the view tab and then clicking zoom out or zoom in. In addition, to pan around the layout, hold down the mouse wheel and drag the mouse accordingly.



Extension: Solid Objects

1. Create a new object by double clicking the layout, this time selecting sprite.
2. Load in an appropriate texture such as a wall, tree or rock.
3. Go to the properties bar and click on behaviours.
4. Select the solid behaviour from the options listed.
5. The object is now solid\*.

\*This can be tested once a playable character has been implemented.

Jamie Evans, Player;

1. Add a new layer on the right-hand side and selecting layers and name this new layer as Main.
2. Insert the mouse and keyboard object as well as 4 Sprites for the player, monster, explosion and the bullet.
3. Select the player sprite, then navigate to the properties bar. Click the Behaviours category.
4. Select 'add behaviour' icon in the behaviours. Select the 8-direction movement.
5. Select Scroll To, this will make the game screen stay with the player. Also, select Bound To.
6. On the Bullet object add the behaviour bullet. Also, add the behaviour destroy out of layout.
7. For the monster sprite, select the bullet behaviour.
8. Add the Fade behaviour to the Explosion object.
9. Click on the Monster object and change the speed to 80.
10. Duplicate a few of the Monster objects by holding control and then click-dragging to another part of the area.
11. Go to the event sheet tab towards the top of the program

The Event Sheet tab.

1. Click on new event and reproduce this event by clicking on system and selecting every tick, then add action and select the player sprite and set the condition like below.

The completed event.

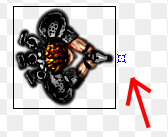
1. Do the same for this

Ghost shooter event 2.

1. In the project/object bar for the player select the edit animation
2. Click on the origin and image point button and click the green + to add an image point.

The origin and image points tool.

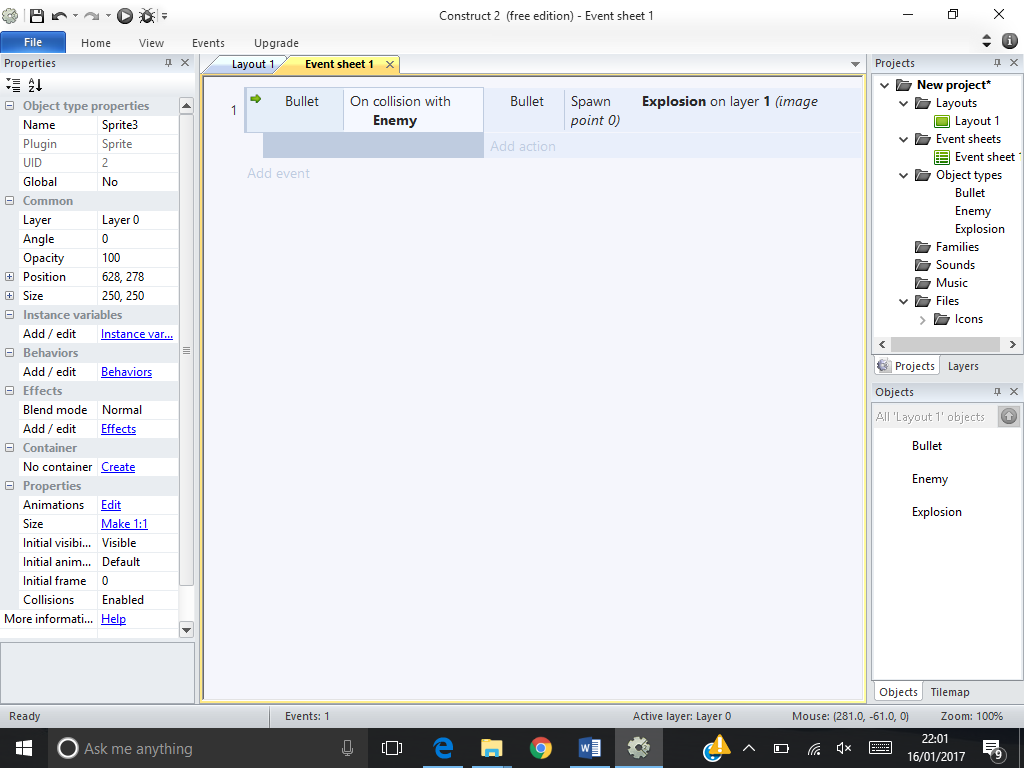
1. Set the image point as shown



1. Once your back in the events tab double click on the shooting event and change the image point to image point 1.

The edited event.

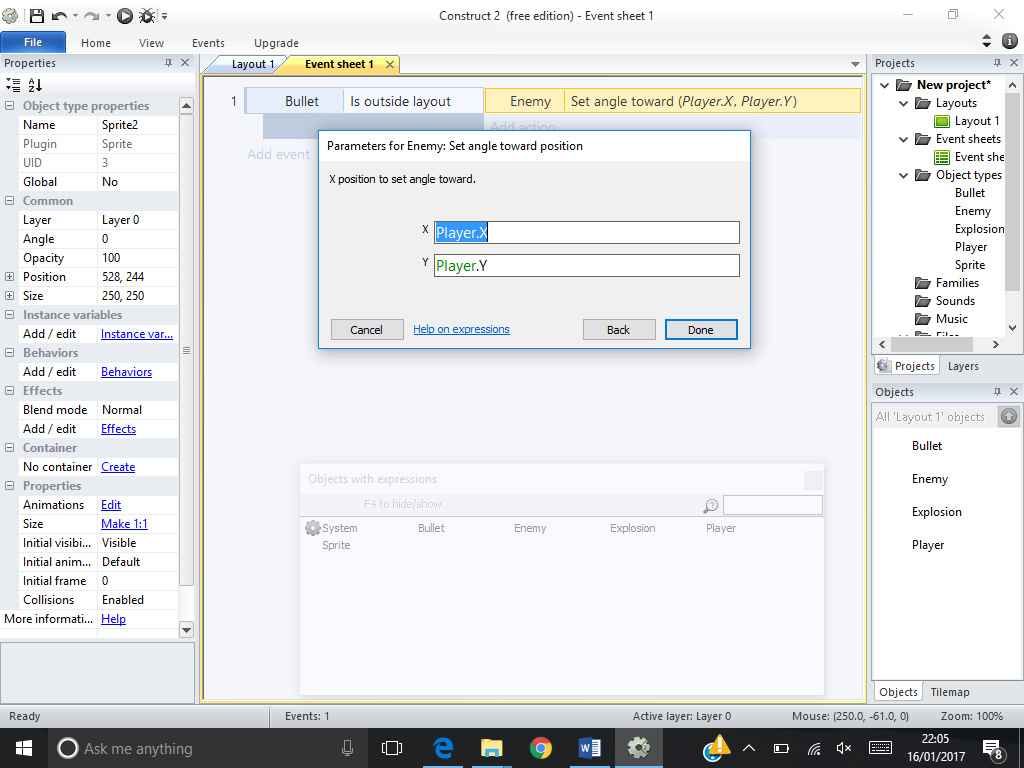
1. Create a new event for destroying the monster.



1. Now create another event to position the monsters at the start.

Event 4 of the tutorial.

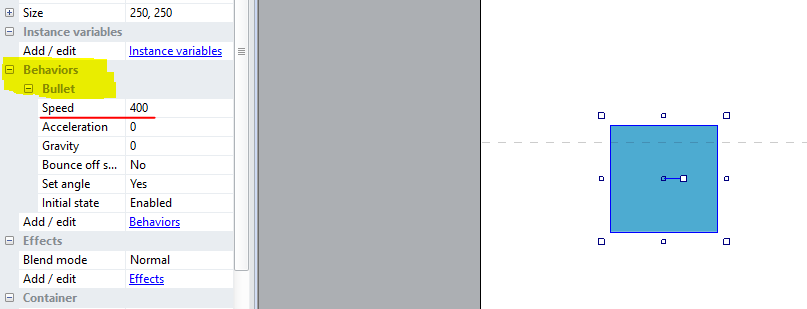
1. Now create a final event for positioning the monster if it goes out of bounds



Vlad Dumitriu, Enemies;

Tutorial:

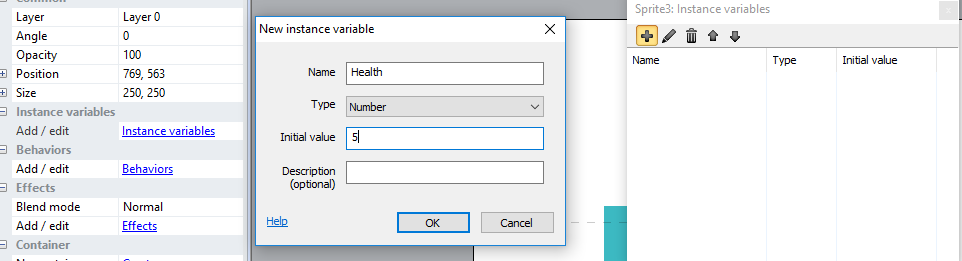
* Create a sprite and press on Add / edit Behaviours to add the bullet behaviour. Now press run and test the program.



Step 1: make another sprite

Step 2: press on add / edit Instance variables

Step 3: make a variable type number with a value and press okay



Step 4: go to events

Step 5: make an event→ select your sprite (the monster) → on collision with another object → select your player

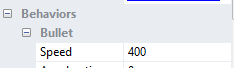
Step 6: add actions → select the player → subtract from → and choose the DMG that will be dealt

C:\Users\vladd\Desktop\Tutorialss\subtract health.PNG

Step 7: make another event → choose the player→ compare instance variable → choose less or equal to 0 → add action → choose the player → destroy

C:\Users\vladd\Desktop\Tutorialss\healthsys.PNG

To change the speed of the monsters you can go to the behaviours and just change the speed value.



Through this module, I have learnt that making games or even teaching games is very difficult to be done alone, this is because the multiple features a game consists of is too many for one person to complete or teach. The different aspects of working as a team were highlighted whilst working individually, we still communicated and referenced each other’s work to ensure that the game would have a consistent theme. Also, we each demonstrated and informed each other how we had used the game engine to create our specific parts. The game engine we used allowed for us to search for solutions to problems and see different people’s perspective and their solution to creating similar features of our game making it a very asset, which in retrospect allowed for us to rely on each other more making us a more effective team for the fact that we could assist on each other’s individual section. As a team, we worked effectively together and supported each other when one would struggle or have a question about the vision of our game and the way to implement that vision. The game engine was able to accommodate our team work by making it so easy to add and alter previous work and have it be changed throughout the project. The module gave us the opportunity to present our work and show what we had learnt through developing the game. When presenting each of us took turns to show what we had created, we did this effectively as a team through assisting the teaching of the section through ensuring the audience could partake and wouldn’t be lost. We could do this because the fact that we could helped each other in creating the game overall and individually. Regarding the tutorials, creating them was a very productive task as it allowed for us to reciprocate what had learnt and could determine what we should improve on or change making the game much better than it already was by researching other tutorials. Another point of making the tutorials which was highlighted was the fact that it meant we could learn each other’s sections and even improve each other’s tutorial making our presentation exceedingly greater. The module within its entirety, allowed for us to expand our knowledge on the multiple game engines, giving us better experience making the course that more fulfilling through an improving environment.

The different game engines that were taught were Flowlab.io, Scratch, Clickteam Fusion, Game Maker and Construct2. The different engines all had a free edition which established limits to what each group could do compared to what they wanted to do. Flowlab.io is a web based game engine making it easy to use and doesn’t require the user to download or constantly update the engine like many of the others do. The engine provides a good number of features like animation a more in-depth event system and transition of different levels. However, as Flowlab.io is still in its beta the problem that is all to present is the lack of tutorials making learn the engine a very difficult task. This engine is very suited to platformers or arcade type games This is because the engine provides a way to introduce the different controls through bundles that make it much easier to implement the events. However, it would be very poor at creating games of the run and gun genre as it doesn’t provide a large enough canvas to stimulate endless running. Scratch is a simple game engine that also uses events that allows for multiple levels and transitions. Scratch has an extensive library of assets and makes programming the game much easier. This is done so through snapping the events together making blocks of code to make a much cleaner and organised engine. However, one problem is the number of blocks of code used to make a basic game such as a platformer, without the tutorials learning this engine could be a daunting task. The engine allows for simple games such a role-playing game like Final Fantasy. This is because through the different events available users are able to create the turn based system and create an effective menu system.

Clickteam Fusion also uses events however implements this through a different approach. There are set properties that you can choose the game object in question will affect or be affected. The engine is comprehensive with frame editor which allows you to be much more precise with what you want the game you are creating to look like and work. Clickteam fusion provides an easy and understandable event sheet that doesn’t particularly need extensive assistance to complete basic tasks like player control. However, this engines events can become over cumbersome and prevents any user from making more complex games. This engine would be more suited to making games that are arcade or basic platformers genres, this is because the engine is better suited for simpler event sheets and if anything, more advanced would cause more problems with event management. Game Maker can be either coded from scratch or a drag and drop menu can be used to create games. The drag and drop method works similarly to events making it simpler for new users to learn the coding bit by bit and then eventually being able to solely make a game from scratch through coding. Game Maker is extremely accommodating for new users and provides an excellent platform for new users who are willing to learn coding instead of just using events like the previously mentioned game engines. Another point is that this engine unlike the other engines can make 3D games not just 2D games. However, Game Maker uses its own language that no other engine facilitates, which doesn’t allow for universality making it pointless to learn if this engine is being used to learn how to programme. This engine would best be used to make 2D games over 3D games as it is more suited to 2D game features like side scrolling.