Discussion of Pipeline & Game Data Files

# Using the file system

When using the file system a couple rules have to be followed.

* After the ‘:’ no spaces should be used if it’s string. “Image:background\_1”
* Vector2 will be loaded from the same line. “Position: 33, 33”
* Boolean will be writing in lower case and have no spaces. “Fade\_particle:true”

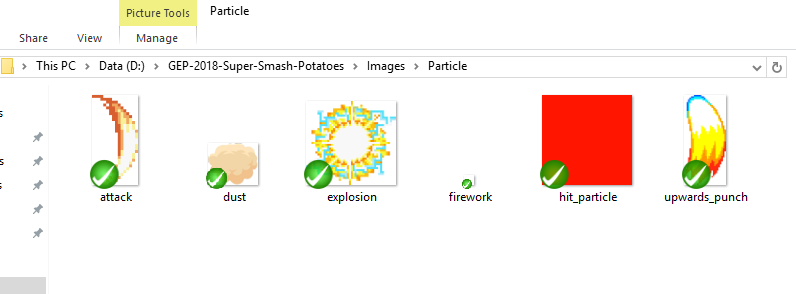
If these rules are followed the file will be read fine.

# Creating a new particle

## File Side

When creating a new particle in our games engine the user will need to do a couple tasks. First will need an image for the particle, they image should be named after the particle.

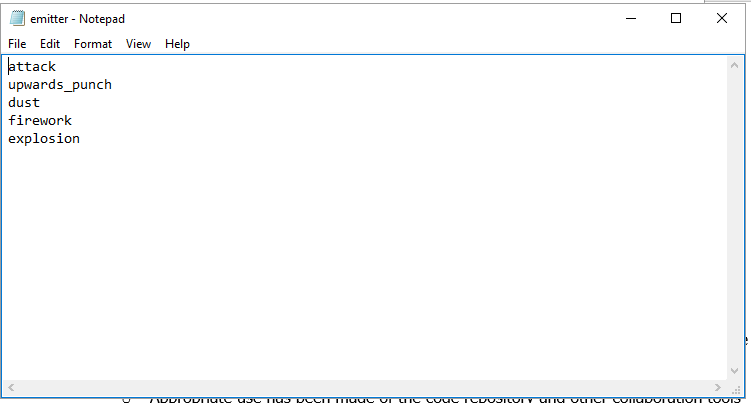
### Example image:



The image is an explosion particle, called explosion.

### Creating a emitter

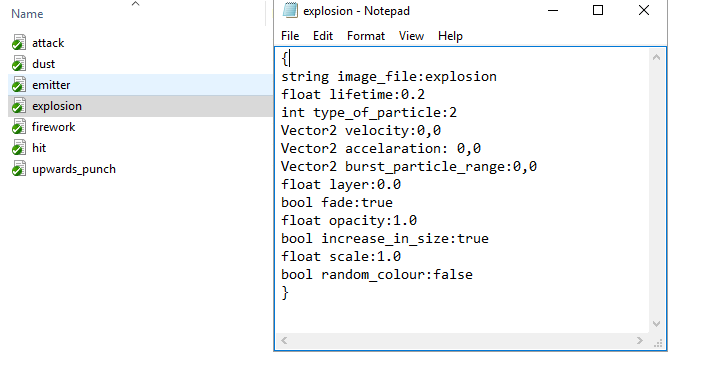
Next the emitter will need to be created, the emitter will need to be named after the particle image. To create a new emitter the user will have to open “Direct3D12 Win32 Game\\particles \\emitter.txt”.



Adding the name to this file will create the emitter system when the game is loaded.

### Particle behaviours file

Each type of particle has its own effect, which determines how the particle behaves. The behaviours will read through a file, the file will be text document named after the emitter name.



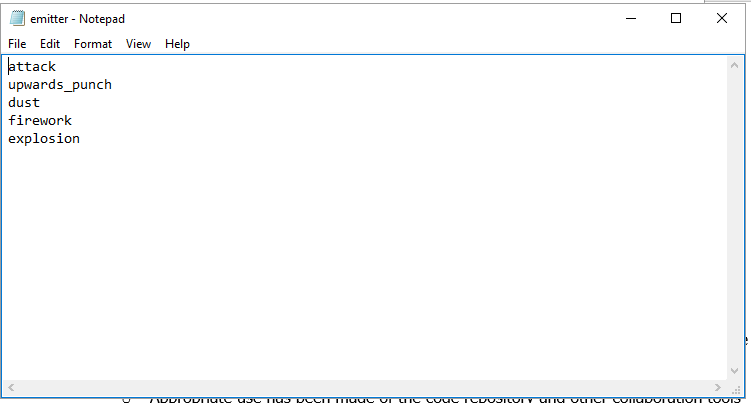
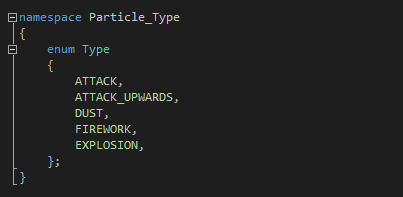
Most of the behaviours are self-explanatory, except for type of particle.

1 = Burst of particles (Based on burst\_particle\_range)

2 = Single direction (Based on the velocity and acceleration)

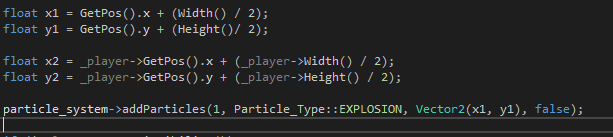
## Engine Side

Now all the files have been created, the user will just need to add a couple lines of code. In the “ParticleSystem.h” file, there is an enum for particle type. They will have to add the name of the particle to that list and save.



**This has to be in the same order as the “emitter.txt” file.**

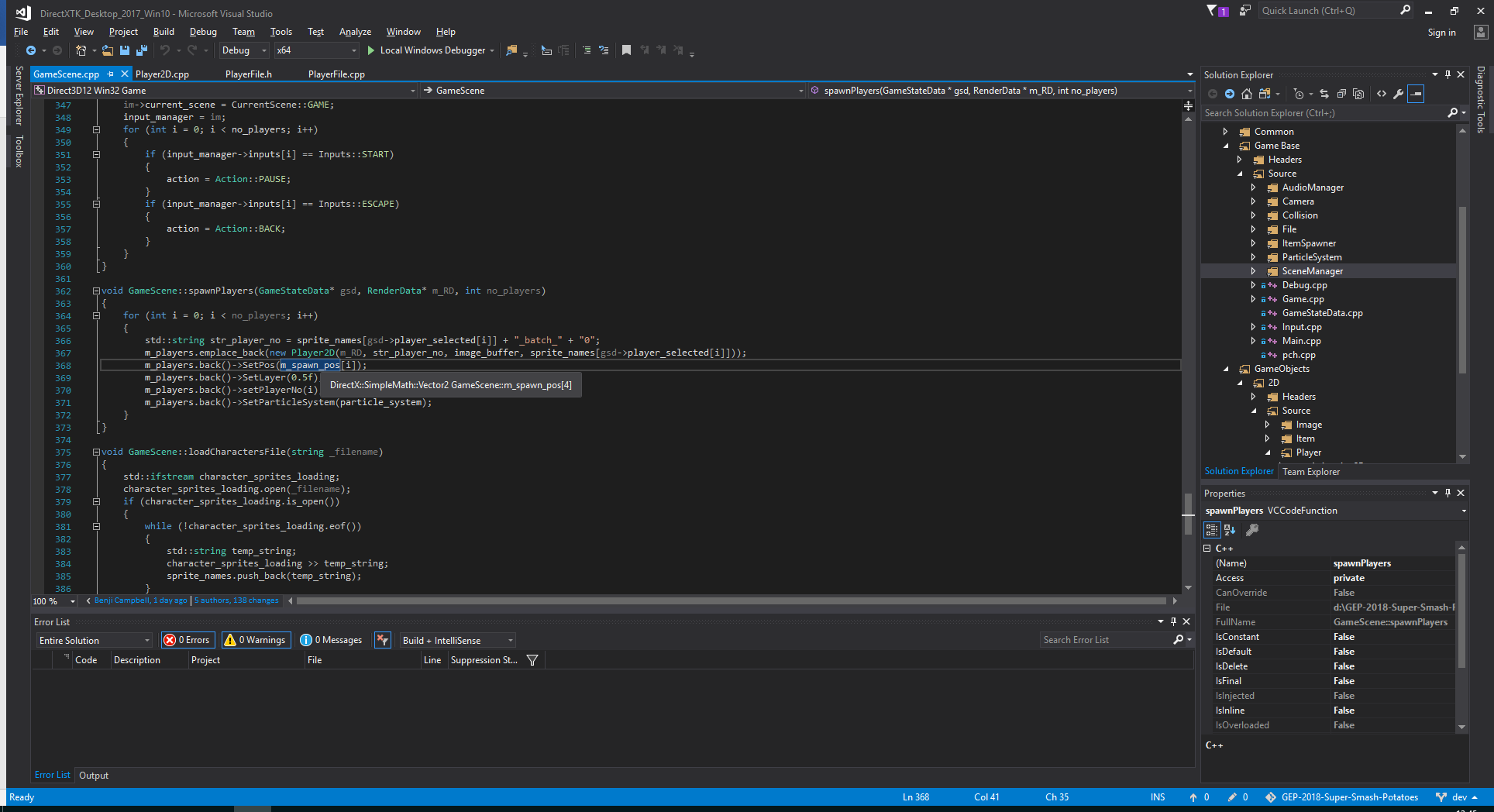
After this the particle is now in the engine, the user will just need to call the “addParticles” function where they want the particle, example below.



Add particles will take in the amount of particles, the type of particle, the position and if the image is flipped or not.

# Creating a Player

In order to create a Player object you need:

* A player stats file
* Player sprites batch file
* Player sprites image

The first parameter is the render data used for drawing the player to the screen.

The second parameter will be the path to the sprite batch used for loading in the ImageGO2D.

The third parameter is the image buffer used to stash all of the images.

The fourth parameter is the link to the stats file of the player.

## Setting up the stats file

The stats file should be setup like this:

**{**

**Jump: 75 - Jump Power [Float]**

**Upwards Punch: 100 - Upward punch jump power [Float]**

**Animation Sheet:kirby\_sprite\_batch.txt - Path to animation sprite sheet [String]**

**Drive: 900.0 - Drive [Float]**

**Drag: 3.0 - Drag [Float]**

**Throw Item(CAPS):Y - Button key code for item throw [int]**

**Jump:A - Button key code for jump [int]**

**Jump down platform:DOWN\_A - Button key code for jump down [int]**

**Upwards Attack:UP\_X - Button key code for up attack [int]**

**Slam:B - Button key code for down attack [int]**

**Attack:X - Button key code for neutral attack [int]**

**}**

## Setting up the sprites and batch files

The sprite batch file should contain a series of x and y position determining the area of each sprite on the sprite sheet similar to this:

**1 1 80 75**

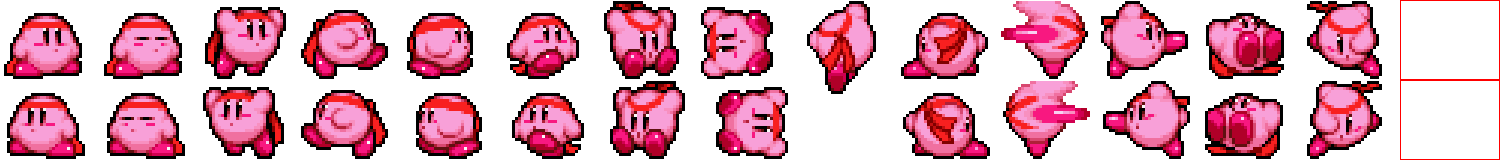
**1 1 88 78**

**101 1 188 78**

**201 1 288 78**

**301 1 388 78**

And the sprite batch image should be setup similar to this:



Where the sprites on the image correspond to the sprite batch file.

Once a player object has been instantiated it can be called via the update and render methods. The scene that contains the player should be responsible for calling the player’s render and update methods.

# Adding a new Background image

Create a new entry in the backgrounds.txt file similar to this:

{

image name:blackground\_1 - Path to the image file

position:1, 1 - Position on screen

origin:1, 1 - Origin position

scale:1, 1 - Scale of the image

orientation:0 - Rotation of the image

layer:1 - The layer number

type:0 - The type of game object

sprite min X & Y:1, 1 - The Rectangle bounds of the sprite

sprite max X & Y:1280, 720 - ``

}

Remember to have the background sprite added as an image as well. You should now be able to specify the new background when creating a level.

# Adding a new object

Create a new entry in the default\_objects.txt file similar to this:

{

image name:platform\_small - Path to the image file

position:340, 500 - Position on screen

origin:1, 1 - Origin position

scale:1, 1 - Scale of the image

orientation:0 - Rotation of the image

layer:0.5 - The layer number

type:1 - The type of game object

sprite min X & Y:1, 1 - The Rectangle bounds of the sprite

sprite max X & Y:150, 48 - ``

}

Remember to add the new sprite to the images directory and the game object should now be available when creating a new level.