**About**

This is a game engine, which is acts as building blocks to easily create any game in java.  
You will need to know the basics of programming in java to continue (variables, methods, loops, etc)

This document contains most of the needed info on how the game engine works and how to use it.

This game engine is heavily based on the *Unity* game engine.  
Most of what applies there will also apply to some extent here.  
Documentation for Unity can be found [here](https://docs.unity3d.com/560/Documentation/Manual/UnityOverview.html)

This game engine is still a work-in-progress  
Which means features will be added later down the line. This game engine is only for 2D (for now?)  
github link can be found [here](https://github.com/NekDuk/Game/)

Each section will follow this format

**Title of section**Section description

|  |  |
| --- | --- |
| variable name *[variable type]* | variable description |
| function name(**ParameterType** parameters) *[return type]* | function description |

**GameObject**Every object in your game is a GameObject. This means that everything you can think of to be in your game has to be a GameObject. However, a GameObject can’t do anything on its own; you need to give it properties before it can become a character, an environment, or a special effect.

A GameObject is a container; you add pieces to the GameObject container to make it into a character, a light, a tree, a sound, or whatever else you would like it to be. Each piece you add is called a component.

Depending on what kind of object you want to create, you add different combinations of components to a GameObject. You can think of a GameObject as an empty cooking pot, and components as different ingredients that make up your recipe of gameplay. This game engine has lots of different built-in component types, and you can also make your own components.

|  |  |
| --- | --- |
| addComponent(**Component** c) *[void]* | Adds a component to the game object |
| name *[String]* | The name of the game object |
| tag *[String]* | \*A **Tag** is a reference word which you can assign to one or more GameObjects. |

\*For example, you might define “Player” Tags for player-controlled characters and an “Enemy” Tag for non-player-controlled characters. You might define items the player can collect in a Scene with a “Collectable” Tag.

Tags help you identify GameObjects for scripting purposes. They ensure you don’t need to manually add GameObjects to a script’s exposed properties using drag and drop, thereby saving time when you are using the same script code in multiple GameObjects.

Note: There are a lot of variables and functions across multiple scripts. The component class implements all the core functions into itself so that it can be directly called without reference.

**Component**  
Components are the nuts & bolts of objects and behaviors in a game. They are the functional pieces of every *GameObject*. If you don’t yet understand the relationship between *Components* and *GameObjects*, read the *GameObjects* page before going any further.

A *GameObject* is a container for many different Components. By default, all *GameObjects* automatically have a *Transform* Component. This is because the *Transform* dictates where the *GameObject* is located, and how it is rotated and scaled. Without a *Transform* Component, the *GameObject* wouldn’t have a location in the world.

|  |  |
| --- | --- |
| transform() *[Transform]* | Returns the transform its object |
| gameObject() *[GameObject]* | Returns its game object |
| update() *[void, @Override]* | Gets called every game tick |
| start() *[void, @Override]* | Gets called once when the its object is first created |
| render(Graphics2D g) *[void, @Override]* | Same as update() but for rendering |
| spawn(**GameObject** object) *[void]*  (**GameObject** object, **Point** position)  (**GameObject** object, **Point** position, **double** rotation)  (**GameObject** object, **Point** position, **double** rotation, **Transform** parent) | Creates a game object at the specified location and rotation |
| getPrefab(String prefabId) *[GameObject]* | Returns a preset game object via the given id |

**Transform**It is impossible to create a *GameObject* in the Editor without a *Transform* Component. This component defines the *GameObject*’s position, rotation, and scale in the game world and Scene view.

|  |  |
| --- | --- |
| position *[Point]* | The position of the transform in x-y coordinates |
| rotation *[Double]* | The rotation of the transform in degrees from -180 to 180 |
|  |  |
|  |  |
|  |  |

**Sprite**Sprites are 2D Graphic objects. Sprites are essentially just standard textures but there are special techniques for combining and managing sprite textures for efficiency and convenience during development.

|  |  |
| --- | --- |
| name *[String]* | The name of the sprite |
| image *[BufferedImage]* | The raw image of the sprite. |
| size *[Dimension]* | Size and scale of the image shown in the game |
| resolution *[Double]* | Determines how big the image will appear. (1x is normal res, 0.5x is half, etc) |

**SpriteRender**er  
Sprites are rendered with a Sprite Renderer. Use it to display images as Sprites.

|  |  |
| --- | --- |
| sprite *[Sprite]* | The sprite to render |
|  |  |

**Camera**-Manages the position and rendering stuff on the screen

|  |  |
| --- | --- |
| getOffset() *[Point]* | The screen offset of the camera |