Eksedra Engine Documentation notes:

* Mainly how to *use* it, not how the engine itself works
* Setting up a project
  + Necessary folders
  + Program.cs & Main function
  + Custom objects (reference to GameObject)
* External Game Resources
  + Images (reference to EksedraSprites)
  + Fonts & SFML Text (reference to SFML)
  + Audio Files & SFML Audio (reference to SFML)
  + Rooms (built off of game objects) (reference to GameObject)
* EksedraSprites vs images (reference to GameObject)
* GameObject abstract class
  + How to inherit
  + Each of the 11 functions + constructor
    - Draw functions & SFML (reference to SFML)
    - Update function and delta time
    - OnCollision and “other”
    - OnKey\* and key event arrays (reference to engine)
  + Motion, persistency, and other builtin variables (reference to GameMaker)
  + Timers
* GameRoom class
  + Explain roomsize
  + Explain gameobject list (reference to GameObject)
  + Explain how file translates into a CSharp
* Overall engine design
  + Each of the 5 threads (update, draw/main, key, collision, timers)
    - Note timer system/countdown (reference to GameObject)
    - Note update and delta time (reference to GameObject)
    - Key states
  + Resource access (reference to external game resources)
  + SFML window and other connections
  + Reference to GameMaker expression
    - Why type list needed
* SFML (link to SFML website)
* GameMaker (link to yoyogame)

GameObject:

The Eksedra Engine is built upon the relationships between different “game objects” within a ![Room](https:// link to room). These game objects are all based on an abstract C# class in the Eksedra namespace called GameObject.

**Methods:**

The abstract methods in GameObject include Init, Draw, EarlyUpdate, Update, LateUpdate, OnKeyUp, OnKeyDown, OnKeyHeld, OnKeyOff, OnCollision, and OnTimer. A child object may also include a constructor which is used to do things such as allow start position placement or set ![persistency](https: link to persistency further down).

**Init**

At the start of the engine, all objects in all rooms and all persistent objects get this method called to initialize them. It is void and takes no arguments.

**Draw**

This method is somewhat strange looking with the parameters it takes:

`public abstract void Draw(RenderTarget target, RenderStates states)`

This method has it’s somewhat strange parameters because it is a piece of the underlying graphics system, ![SFML](link to SFML page). This method is what allows the Window to draw our object. Naturally, in this method, you use ![SFML](link to SFML page) draw code to draw your sprite or text or primitives or whatever else you want to put on the screen.

As seen in the ‘test’ example, this is often used to draw the ![SpriteIndex](link to spriteindex doc) with `target.Draw(SpriteIndex);`

**EarlyUpdate, Update, and LateUpdate**

This function is run over and over again for all objects in the current room. EarlyUpdate is called for all objects first, then Update for all objects, finally LateUpdate for all objects. Each method takes the parameter “deltaTime” which is the time passed since the last update.

Note that at the end of LateUpdate, the object’s position is updated based on its ![horizontal speed](link to HSpeed) and its ![vertical speed](link to VSpeed).

**Key Functions**

OnKeyDown triggers as soon as a key is pressed, where OnKeyHeld triggers when a key is held down for longer than a single press.

OnKeyUp and OnKeyOff parallel them, triggering as soon as a key is let up and after a key has been off for longer than a single release.

These take a keyState parameter which is a large array booleans corresponding to all ![SFML](link to SFML page) Keys possible.

**OnCollision**

OnCollision triggers when the object collides with another, that is to say that the current objects ![mask](link to collision masks) intersects another’s. The parameter is the other object that the object collided with.

**OnTimer**

The game object contains 10 floats in an array called “Timers.” When one of these timers is set to a positive value, it will count down until it reaches 0, at which point this method will be called.

**Parameters:**

Each game object has several built-in parameters: