# Engine

#### Goal

* Make Sidescroller engine
* Make level editor

#### Source code

* GPAT game code (+ explanation in book)
* GCC code
* GPP (Design Pattern)
* UE4 source code
* Resource folder
* Coco structure

### Concept to see

* Multithreading
* Data oriented design
* ECS
* Precompiled header

## Task

### Ongoing

#### Quick task

##### To do

* Recap and reset focus

##### Tasks

* Make namespace “ engine” first, then add more if needed
* Watch/read : finite state , cherno channel, code example
* Game State
* Ressource Manager : factory pattern, object pool,
* Actor : data access between components and owner ? see ue4 , implement a world position (instead of formule to get center ) with 0.0 at center of screen
* Precompiled header for external and non changing import

### Recap

* Architecture
* Resource manager
* Input
* Event handler System
* Graphics
* Ray casting
* Math Library
* Camera
* Physics engine
* Save system
* Scripting language
* Data file
* Framework
* UI
* Editor
* Animation
* Text input handling
* Loop&Time&Framerate
* AI
* Sound
* Optimization
* Other

### Systems

#### Architecture Design

* World Class
  + Contains : sounds, fonts, textures

##### GameState Pattern

* Create a StateClass
  + With handle input(), enter(), exit(), update() ,
* Create a gameinstance
  + Contains state of the game, …
  + See finite state machine for game state
  + Use hierarchical state pattern ?
* Game Instance class
  + Contain the current gamestate
  + Contain world resource ?
* Resources
  + Onetab webpage
  + Resource folder
  + GPAT p 196

##### Other

* Entity based engine vs inheritance based engine
  + See GPP
  + See cowboy prog
* Object class
  + Actor : base class for trigger ect
    - Character

#### Ressource Manager

##### Idea

* Initialize member in constructor ?
  + Texture in sprite for example ?
* shared/unique pointers
  + Use raiiwraper ?
  + See item 14 of effective c++
* Figure out how to architect an AssetManager ( where ? how ? )
* Does objet like button or sprite use composition for texture or store all texture/sound ect in a global structure ? Look at design pattern
  + Look at GCC
* See Factory pattern
  + For sprite, …
* See Service Locator Pattern
* Use Hashed of filename instead of direct file name ( use xml )
* Map for each type of asset ( texture, button, sound, ect)
* with object associate with name ( name is the key )
* name have a specific nomenclature : L1\_2 for level 1 part 2 for example
* Resource : sprite, font, actor …

##### Implementation

#### Input

* Implement Command Pattern
* See Chap 5 of GPAT

##### Keyboard event

* + See tuto05
  + Implement the Array of bool for each key system

#### Event\_handler System

* Implement event similar to delegate UE4 system

#### Graphics

##### Rendering

* Develop a system to render only sprite whe needed, and destroy them automatically
* Develop a world coordinate system with 0.0 at the center of the screen ( width/2 . height / 2)
* Use Viewport method to render minimap or other
* See 10 for rendering multiple textures
* See 11 for rendering part of spritesheet
* See 12/13 for color/alpha modification rendering
* See 15 for flipping/rotating

##### Tiling

* See 39
* See p35-39 of GPTA

##### Scrolling

* See 30,31
* See GPTA 30-35

##### Animate sprite

* + See 14
  + GPAT 25 – 28
  + Implement functionality to use array of texture

#### Ray Casting system

#### Camera

#### Physics engine

* See GPAT p129
* See Box2D
* Line Trace

##### Collision detection

* Choose a method ( AABB, collision sphere ect )
* See 27,28,29

##### Movements

#### Save System

#### Scripting Language

* Try to implement Lua

#### Data file

* See xml, json , other

#### Framework

* Make Actor class
* Use controller system ?

#### UI

* See GPAT chapter UI
  + Implement a stack system

##### Button Class

* Add font texture as attribute + set color/alpha method
* Clean class

#### Level/Tile editor

* Use Qt

#### Animation System

* Use FSM
  + Should be stack
  + Should it be hierarchical ?

#### Text input handling

* See 32

#### Loop&time&FrameRate

* + See chapter 1 of GPAT and 24,25
  + See 25 and GPAT p12 for capping FR

##### Game loop

* Try To implement multithreaded game loop

#### AI

* See GPAT chapter AI
* Implement FSM System
* Try to implement Behavior tree system

#### Sound

* + See 21
  + See GPAT p 120

##### DPS Effects

#### Optimization

* Precompiled header

#### Other

* Add an option to show FPS

##### Joystick Direction

* See 19