Resources

<https://emielongamedev.wordpress.com/writing-a-game-engine-from-scratch/>

<http://gamedevelopment.tutsplus.com/tutorials/lets-build-a-3d-graphics-engine-points-vectors-and-basic-concepts--gamedev-8143>

good inspiration: <http://www.grandmaster.nu/blog/?page_id=118>

This is a video game.

It is solo, multiplayer, loot based, level based, partial class based, skill usage based, real time action game. There is lots of randomization.

Solo:

It is a primarily solo player game, with incentive to be multiplayer. Single player progression is strong.

Multiplayer:

There is a multiplayer aspect, which includes trading. Players can group up and fight, increasing game difficulty.

Loot based:

All loot is randomly generated from a loot table.

Level based:

Players gain levels, which can be used on skills and increases abilities and stats.

Partial class based:

There a multitude of classes to choose from, and they have the ability to spec any amount of points into any class, limited by a minor progression tree for each class skill growth. Players should be able to spec into a multitude of classes with a bit of specialization in order to create a unique playing character.

Skill usage based:

Players use skills which move the player character, deal damage, and a variety of other abilities. While the player can auto attack, it is not enough to carry the player forward. Building and customizing the skill with be a core part of the customization of the game.

Real time action:

The game is going to take place in real time. Monsters move and attack without waiting for player input.

Increasing game difficulty:

The player has the ability to adjust the difficulty of the game in order to gain better rewards, testing themselves at a risk reward balancing game.

Randomization:

Skills, items, enemies, terrain, and just about everything will be randomized.

The engine needs to be multithreaded, 60+fps, multiplatform, 3d, procedural graphics, have physics, and scripted.

Multithreaded:

The game will be able to take advantage of multiple cores to process events.

60+ fps:

The game must run smoothly on a decent computer.

Multiplatform:

Preferably, can run on multiple different OSes – windows mac and linux preferably

Procedural graphics:

The terrain/map will be randomly generated.

Physics:

Actions and events will have body/framework responses

Scripted:

It will be extensible with a programming language.

The game engine will be composed of thus:

Sound engine

Input engine

Physics engine

Graphics/rendering engine

Networking engine

Logging engine

Game logic engine

Gui engine

Scripting engine

Following gamasutra and general good coding practice, this is my attempt to put the engine in lasagna code and a structured layered architecture

Game logic - gui engine

Interaction layer – scripting – logging engine

Sound – input – physics – graphics – networking

Game logic interacts with logging and scripting to mess around with objects. The lowest layer has an interaction layer between it and game logic/gui to display and request object interations. All intractions are logged. Scripting modifies the lower level, as well as giving ability for game logic to manipulate items.

The interaction layer basically takes all the spaghetti code and crazy dependencies and centralizes them. Interaction signals a message log.

Game logic has the following

Game logic

Objects

Agents

NPC

PC

Pet

Map

Non agents

Structure

Items

Projectiles

Particles

Camera

Generation

Map generation

Item generation

Etc. generations

Event

Dialogue

Action

Database manip

Object Manip

^obviously not complete

Preliminary:

I need to create the project folde.r This will be structured up into various folders/components, such as distinguishing engine core from game data to utilities and resources and to actual game execution.

I need to create my game development tools. This means

* Code benchmarking tools
* Ide set up and development
* Setting up wrappers for the languages
* Setting up the data nmanipulation – this involves connecting various components of the logic to distinguish client/server models, how to access and use game data/resources, where it’s located.
* Gui development – mapping gui, database manipulation gui, scripting engine, screens, basically making all the tools through which I’ll manipulate the game.
* Api/jtests and extraneous tools to mess around with.

I’ll also need to update all of my logics into the actual appropriate game logic (such as entity component model)

What kind of guis do I have to create?

* Modeling/animation gui for models, agents, particles, physics, etc.
* Map editing/generation gui
* Database management gui
* Scripting gui (I shouldn’t be making my own ide)
* Messaging/debugging gui
* Gui `ww
* Event management gui
* Resource management/manipulation gui

So, folder structure –

Game engine

Software libraries

Self created guis, runtimes, executables

Game data

Game resources

Game metadata/info

Screenshots

log

configurations

//

Game animations/input as directions?

//

At the end of this is the game documentation and technical design documents, and etcetera

Which ones do I need?

Api and api management document

Exceptions specification

Game engine technical specifications documentation

Architecture/design/requirements

Script documents

Game engine usage and library documentation

Game design and balance (aka, the official game doc)

Game resources documentation

Testing documentation

Milestones document