EpicRay - Design Document

Jonathan Hale

December 29, 2012

1 Introduction

EpicRay is a modern Raycasting engine in Java. I tried to desing it for great versatility. EpicRay should offer most things as Interfaces/abstract classes and always one Implementation. See Renderer for example.

2 Definitions

Representation everything concerning the graphics and audio of the game.
Logic what works in the background, invisible to the user. Stuff like entities, world and physics.
Mechanics special functionalities which can define a game.

3 Resource Management

How resources needed for representation are handled in EpicRay. Resources are the following.

3.1 Resources

3.1.1 Bitmap

Bitmap stores pixel data in an array.

3.1.2 Texture

A Texture is a area on a bitmap which can be used to draw on walls, ceilings, floors, or sprites.

3.1.3 Sound

Sound is a sample that can be played.

3.2 Resource Manager

Resource Manager handles loading and unloading of the resources. It also containes lists of all the existing resources and can be optained through the Resource Manager at any time.

4 Game Logic

4.1 Entity

Represents a *thing* in the game like moving player, enemy or static destroyable barrel, that actually have a function. Purely eastetic things should be included in the game as sprites. As soon as it supports collision it isn't purely eastetic anymore, though.

4.1.1 Player

Just another entity controlled by the Game Handler.

4.2 Tile

A Block in the tile map. It containes a list of references to the entities curently in the tile and a list of decorating sprites. Also has a wall, ceiling and floor texture. Tiles can be opaque (blocks a ray completely, for transparent, or half transparent, opaque should be off) and solid (meaning entities collide with it).

4.3 Tile Map

An array of Tiles. Has getter and setter funktions for the tiles.

4.4 World

Contains everything that makes up the game-world. This means: entities, tilemap, background (?), and so on. No GUI or HUD.

4.5 Input Handler

Handles input such as mouse, keyboard and joystick/gamepad. Input Handler also acts as an abstraction layer for the input ("Joypad").

4.6 Game Handler

Handles input-world-interaction and the HUD/GUI. The class doing this is also responsible for rendering and updating the world.

5 Rendering

Renderers are defined through an abstract class. One implementation is given in EpicRay.

5.1 Technical

Since EpicRay uses raycasting for graphics, rendering is pretty strictly defined already.

Following rendering order must be kept:

1. draw sky/background.
2. draw world (walls/floor/ceiling)
3. draw sprites
4. apply filters.

I was thinking about creating a depth buffer while rendering world and sprites. This would allow depth-dependant filters like DOF and would make Occlusion of sprites and walls alot easier.

Another thought was to create something like a Ray class for the EpicRayRenderer which stores a vertical stripe of pixels and calculates itself and then gets drawn to the rendering destiny bitmap.

5.2 Raycasting

Add a brief explanation of Raycasting here...

5.3 Renderer

The Renderer has a funktion to render a GameHandler object.