OpenRA

website: http://www.openra.net
students: Mariane Raad and Amjad AbdelRahman.

What is OpenRA?

A Libre/Free Real Time Strategy game engine supporting early Westwood classics.

They include recreations of C&C (Tiberian Dawn), C&C: Red Alert, and Dune 2000. These are not intended to be perfect copies, but instead combine the classic gameplay of the originals with modern improvements such as unit veterancy and the fog of war.

OpenRA's primary focus is cross-platform multiplayer between Windows, OS X, and Linux; however, we include a number of single-player missions, and also support skirmish games against AI bots.

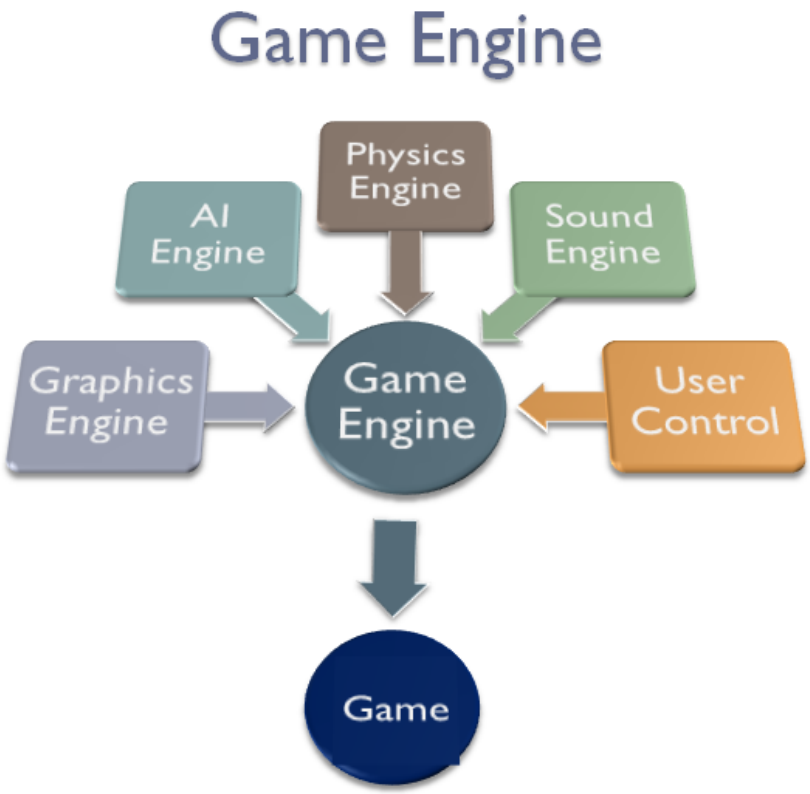
How does it work?

The OpenRA team is a group of developers from all over the world, contributing their time to build the best open-source RTS games (Real-time strategy).

It uses the hardware acceleration of modern video cards using OpenGL and OpenAL.

The whole point was to design a game engine that will support the goal of the team; to develop those nostalgic games in a modern flexible way.

The code is very similar to the game engine which looks like this:



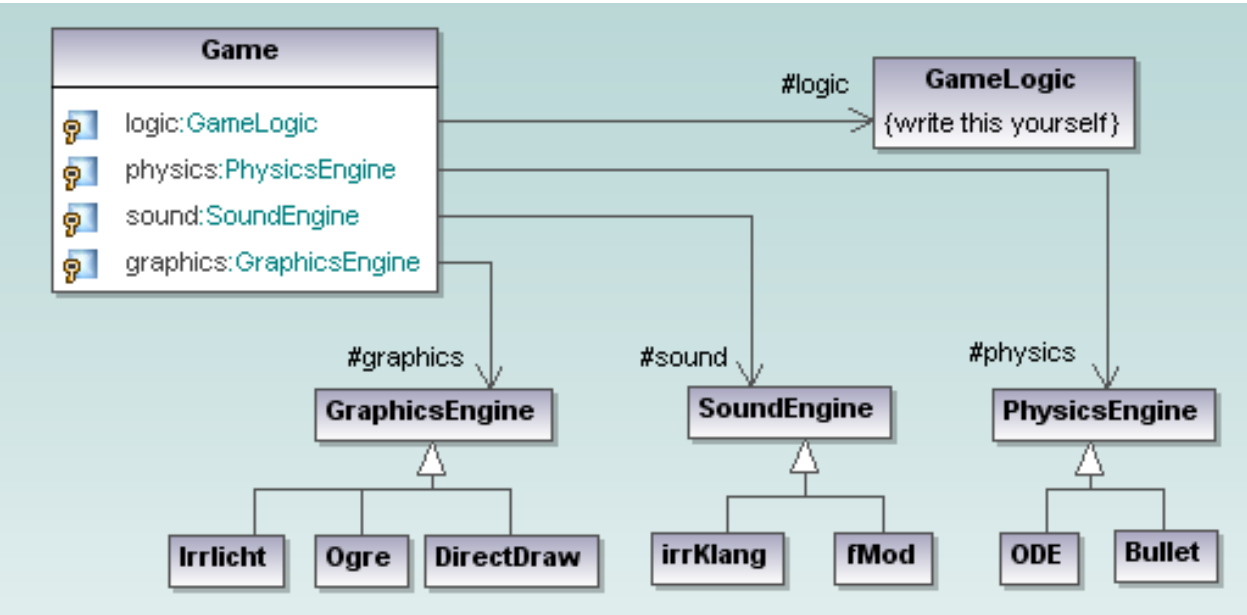
A game engine is a software framework designed for the creation and development of video games. Video game developers use them to create games for video game consoles, mobile devices and personal computers.

components of the framework:

1. Main game program: The actual game logic has of course to be implemented by some algorithms. It is distinct from any rendering, sound or input work.
2. Rendering engine: The rendering engine does the rendering via the chosen method (in OpenRA they use OpenGL, OpenAL). (rendering: is the final process of creating the actual 2D image or animation from the prepared scene. / OpenGL: (Open Graphics Library) is a cross-language, multi-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering. / OpenAL: (Open Audio Library) is a cross-platform audio application programming interface (API). It is designed for efficient rendering of multichannel three-dimensional positional audio. Its API style and conventions deliberately resemble those of OpenGL.)
3. Audio engine: The audio engine is the componentry which consists of any algorithms related to sound.
4. Physics engine: The physics engine is responsible for giving the application a realistic sense of the laws of physics in the application.
5. Artificial intelligence: The A.I. is usually outsourced from the main game program into some special module to be designed and written by software engineers with specialist knowledge.

Some more helpful UML diagrams to show what they are doing:

class diagram:



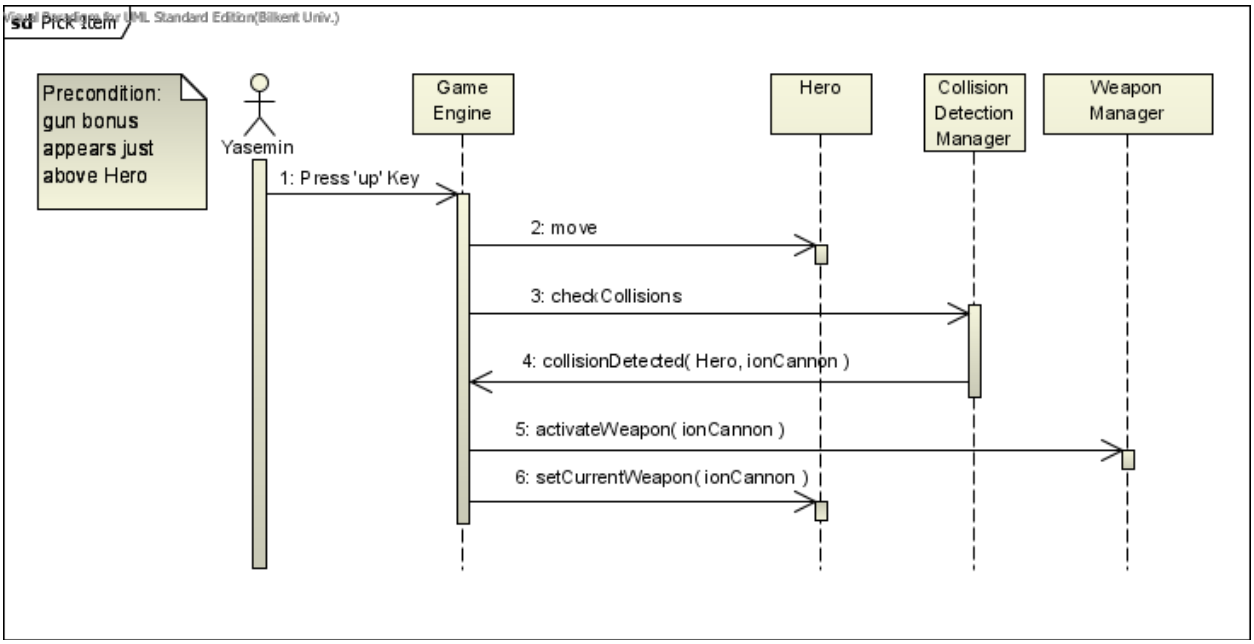
explanation:

the Game is as mentioned previously decided into 4+ sections in this case (logic, physics, sound, graphics).

the two classes (ODE and Bullet) inherit all the functions of PhysicsEngine class, for example the Bullet class is supposed to provide all the movements of the bullet for example that the bullet always movies in a straight line etc.

Same goes for all the sub classes for GraphicsEngine and SoundEngine.

Sequence Diagram:



explanation:

in this sequence diagram, the user presses the up key. First we need to get the appropriate movement look from the Hero Class that belongs to the Graphics/Rendering class. Next we check if we collide with any other object that is there, this exists in the Collision Detection Manager which is part of the Game Logic and so on. The last action is summoning the weapon if needed according to the logic.

Conclusion:

Even though they did not have clear development plans other than posting needs and working on them. Their plan worked for them, they are still today developing more and more features and communicating with the users. They give the users an opportunity to help build and create more packages and features. Team work and github's project management has helped the OpenRA team to stay on top of things, connected and organized.