Sparky Game Engine - Notes

Date: 3/7/2015

Sparky Engine

These notes capture the development of the Sparky Game Engine. The engine will be developed as part of a new video series hosted at the following urls:

- https://www.youtube.com/channel/UCQ-W1KE9EYfdxhL6S4twUNw - you tube videos

Figure 1 - TheCherno youtube home page

- http://www.twitch.tv/thecherno - live videos of the game being developed
- https://github.com/TheCherno - GitHub location of source code for this project

The game engine will be built in 45-days in preparation for the upcoming Ludem Dare competition.

What is Ludem Dare?

Ludum Dare

Developers from around the world get together several times a year to take a weekend to build a game from scratch using the game theme presented to all the developers. The game theme is proposed and selected from the game development community. The goal is for the individual or a team to create a game in 72 hours. You can use any tools or libraries or start with any base-code you have. You can even use art work and music from 3rd party sources. You are encouraged to create games that can be played right in the browser. The source code created is made freely available to community. This practice encourages sharing and growth as others can learn new techniques and programming styles by reviewing code (of the winners and losers).

I will post the latest notes on github at https://github.com/nyguerrillagirl/Sparky-Engine. The repository will contain:

- The latest copy of these notes
- A file named codeCommitSessions listing the commit codes corresponding to the code developed for the end of a video coding session. For example if you want to obtain the code as it

was developed up to video session #5 then the commit number (e.g. d42371e6a7) will represent the code completed up to that session. So checking out that commit[1] will get you to that location in these notes and on the video.

- The code related to this project

I highly recommend that you follow TheCherno on the various locations listed above for more information and to view the videos and to donate.

You can follow me at @nyguerrillagirl on twitter or visit my web site www.brainycode.com . Do not hesitate to provide corrections and suggestions to my e-mail address at nyguerrillagirl at brainycode dot com.

What is a Game Engine?

A game engine is a code package, framework or application tool set that implement the common tasks that all games perform – rendering the screen, physics, input, and collision detection so that the game developer can concentrate on game specific details like art work, and specific game logic.

A typical game engine provides an API or SDK (collection of libraries)[2] that the game developer will use as a starting point to build their own game.

You are probably familiar with popular and well-known game engines such as Unity (http://unity3d.com/) or the Unreal Engine (https://www.unrealengine.com/). These engines are rather complex and have a high learning curve. There are simpler game engines such as GameMaker:Studio (http://www.yoyogames.com/studio) and Construt 2 (https://www.scirra.com/construct2) that make quite easy to build 2D games.

Using a game engine (most are free) will provide an insight into what you want your own game engine to do.

Why build your own game engine?

The benefits of developing your own game engine is the appreciation you gain on what a decent game engine is doing and the technical complexity in implementing one. In addition, it will make is easier to learn and work with more complex game engines.

What Programming Language to use?

The first question being debated is what programming language to use. The two top choices in order to achieve maximum portability is Java or C++.

C or C++ has been the workhorse or primary language of choice for most in-house game engines and/or games. The advantages of using the programming language C++ are:

[1] See Appendix A for a quick tutorial on using GitHub.
[2] http://www.gamecareerguide.com/features/529/what_is_a_game_.php

- Compiles into native machine code so the game engine will be fast
- Supported on all platforms
- Free IDEs and compilers are available (that run on all platforms)
- C++ is well-known by most developers and students
- C++ has many supporting libraries and frameworks

The downside of using C++ is that we probably will not get as many features as we would like completed for our game engine. The reason most game engine developers select C++ is because C++ has been the dominant language choice for many years (yes - use it because everyone else has!). The leading game consoles development tools use C++, many middleware packages are written in C++ and when you think about it if you want support, assistance or want to expand you work of art it will be easier to find C++ experts.

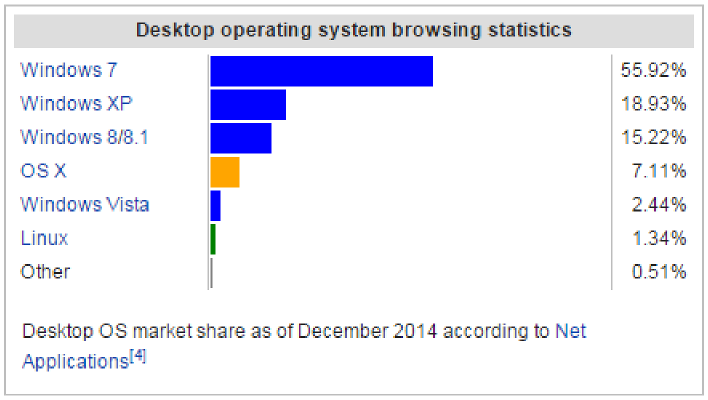
The major downside of using C++ is the plain ugliness[3] of the code since it object-oriented concepts layered on top of C.

Java is up for consideration because

- it is easier to build a cross-platform game engine with it
- software development and debugging is faster
- has freely available tools and frameworks

What platforms should it work on?

The main platform that the game engine will be designed for is Windows. A secondary goal will be to have it run on Mac and Linux environments.



Windows 7	55.92%
Windows XP	18.93%
Windows 8/8.1	15.22%
OS X	7.11%
Windows Vista	2.44%
Linux	1.34%
Other	0.51%

Desktop OS market share as of December 2014 according to Net Applications[4]

Figure 2 - Operating systems on desktops

[3] This is rather subjective since beauty is the eye of the beholder (and therefore so is ugliness). I think it is fair to state that the more experience one has with C++ the less ugly it gets.

As you can see Windows 7 has the largest share of users on desktop environments. The game engine will be built on a Windows OS. I will be using Windows 7.

I plan on using a compiler and IDE that is available on Mac and Linux platforms. I plan on setting up my Mac and Linux machines with the same version of all the tools and provide any notes on differences in one of the appendices.

What are the goals?

The goal is to build a 3D game engine. We will get to our goal by first focusing on building a 2D game engine first.

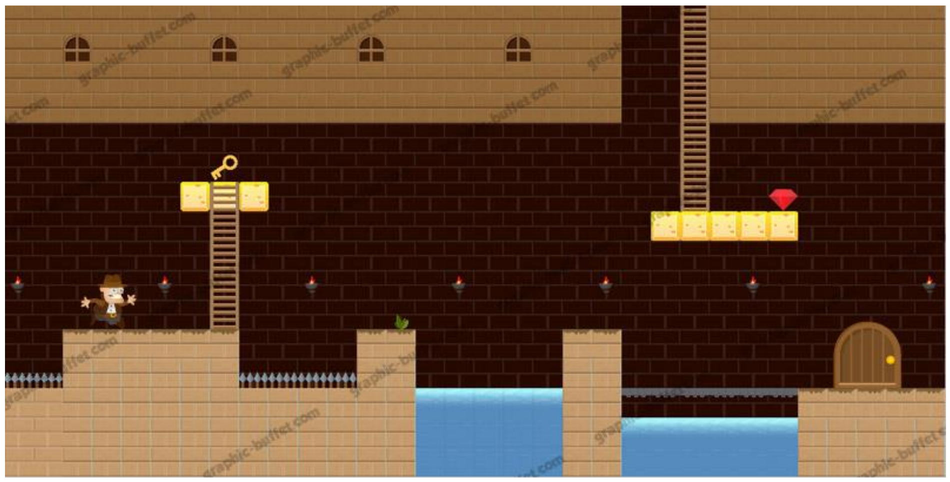


Figure 3 - 2D game look-and-feel

The classic and most popular type of 2D game is the 2D platform scroller (as shown above). The game engine will support the basic elements of building a 2D game:

- Creating and laying out game objects (players and enemies)
- Collision detection
- Input handing
- Scrolling
- Level transition
- Sound
- Heads up display

Most game engines (e.g. Unity) allow you to build either a 2D or 3D game.

Figure 4 - 3D Game, Off-Road Velociraptor Safari

It probably will be stretch but the ultimate goal is to have our game engine support the building of a 2D or 3D game.

What technologies?

The graphics for our game engine will be rendered using OpenGL. For a quick introduction to OpenGL see Appendix C.

OpenGL stands for Open Graphics Library. It is an API (set of libraries) for defining 2D and 3D graphics images.

What languages can developers use to build their games?

We would like not to limit game designers or developers to using the same programming language we selected for the game engine. If we select C++ as the programming language for the game engine it will still be possible for game developers to use Java, C#, C++ or other languages.

In addition, it is quite possible to make our game engine extendable by providing support for LUA or JavaScript as an internal scripting language.

Open Source – Available on GITHub

All material related to this project (even these notes) will be open source and available on GitHub. In addition, we will stick to tools and libraries that are open source and freely available.

What about images and art work?

In order to test and exercise the portions of the game engine we will need to build and "see" things working. This will require game assets such as sprites and texture models. We will stick to freely available assets but may on occasion demonstrate a feature using art assets that we have purchased. I find the website https://www.gamedevmarket.net/ a good place to obtain decently and well-priced art assets.

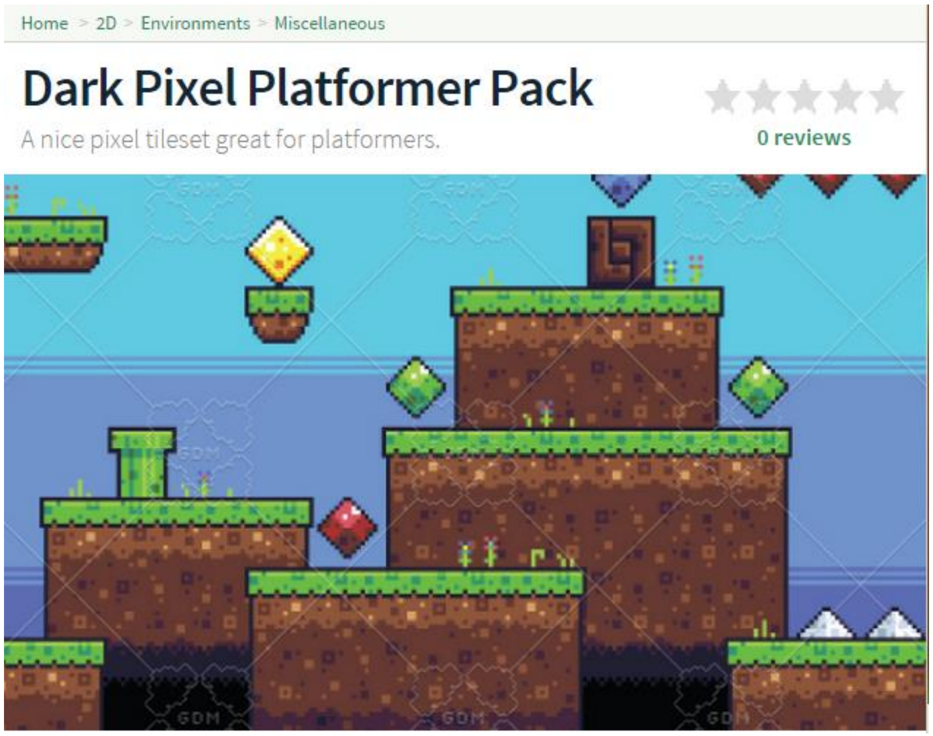

Home > 2D > Environments > Miscellaneous
Dark Pixel Platformer Pack
A nice pixel tileset great for platformers.
0 reviews

Figure 5 - My purchased art assets

I cannot make any purchased art assets available but I do encourage you to consider making a purchase and supporting the game art development community.

Appendix A – Using GitHub

Appendix B – Installing a C++ Development Environment

Appendix C – A Quick Introduction to OpenGL