**HOW TO DO THINGS**

This particular game engine is quite bare bones, but lets do a walkthrough of each java class.

**DISPLAY CLASS**

This class creates a JFrame for the window with the width, height, and title specified in this file. They are constants. It also creates a modified version of JPanel which is used to draw in and creates a KeyListener which will allow us to use keyboard input.

Not much of this class is needed except the getImageGraphics() method and the getImage() method.

getImageGraphics() returns a Graphics2D which we use to draw stuff.

getImage() returns a BufferedImage. This image is the one that will be rendered to the panel.

We modify the display’s image using the Graphics2D object from getImageGrpahics().

**ENGINE CLASS**

This class runs the game, and keeps track of game objects.

By default, this class clears the window’s image before drawing anything to it. This is necessary and shouldn’t be changed.

The game will run on a loop that will call all of the update methods of the objects that it has to keep track of, and then call all of the draw methods of those objects. It will then wait a certain amount of time before repeating so that it updates the window consistently.

Use the add( parentGameObject o ) method to add a object to the engine’s list of objects to update and draw.

Use the get( int index ) method to get an object from the engine’s list of objects.

The size() method returns the size of the list of objects.

By default, the engine sorts the objects in it’s list by a depth value. Lower depth objects are drawn first so they will appear behind objects with a higher depth.

**extPanel CLASS**

This class extends JPanel so that it can override a paint method. This class is not necessary to know anything about so lets skip it.

**INPUT CLASS**

This class keeps track of keyboard input and allows it to work in the way we expect it to in gaming. There are 3 methods that you need.

getKeyDown( int key ) will determine if the specified key **is being held down**.

getKeyPressed( int key ) will determine if the specified key **was just Pressed**. This only last for one frame.

getKeyReleased( int key ) will determine if the specified key **was just Released**. This only last for one frame.

You can use a char in place of an int to represent the key you want to check. For a key that can not be easily represented by a char, you should look up the asci table for the key you care for.

**ParentGameObject CLASS**

This is by far one of the most important classes.

The engines object list will only accept objects that have extended parentGameObject.

You **can not** create a parentGameObject. This class is an **Abstract** **Class.**

An Abstract Class must be extends to a sub class.

Along with having to extend from this class, you have to override the abstract methods which are:

**public void update()**

**{**

**}**

**AND**

**public void draw()**

**{**

**}**

These methods will be called by the engine.

If you do not override these methods, the compiler will give you an error.

This class also has some additional variables that you do not have to use. The only variable that you absolutely need is the depth. The x and y variables are just nice to have.

When you create an object from a class that has extended parentGameObject, it is a good idea to set a depth value. It can be changed at anytime and the engine will reflect those changes accordingly.

**OTHER STUFF**

No methods for collision have been implemented. That should be focused on later. For now, focus on making sure that you understand how the important front end parts of the engine work.

The front end referring to extending the parentGameObject, adding objects to the engine, using the Input class for keyboard input, and using the getImageGraphics() method from the Display class to draw stuff.

As for imports, you should import these classes. These will cover most of the things you will need Except sound.

import java.util.\*;

import java.awt.\*;

import java.awt.image.\*;

The classes testObject1 and testObject2 are examples on how to extend from parentGameObject and use most of the things that were talked about.

MainGame is where we initialize the engine stuff, add some objects to the engine’s object list, and run the game. This class has the **main** method. Use these 3 classes as examples on how the engine works.

By the by, it runs a 60 fps. This can be changed in the engine class by changing the constant variable FPS\_CAP.

LAST THING

Do not remove the files from this folder. Keep them in there so that they are easily orgainized. A java setup file will be packed in that you can use. You do not need that java setup though as it is very similar to the java setup we normally use. Instead of moving to the folder user, move to the folder where this stuff is stored which is call GroupProject

The folder GroupProject should be in the root of the USB Drive. This is where the user folder, java folder, notepad++ folder, and data folder are. Not inside any of them.