Introduction To Rigid Body Physics

Module 0: The Development Environment

University of Reddit
Instructor: Brian Dolhansky
Course website: http://universityofreddit.com/v2/class.php?id=111
Course email: intro.to.rbp@gmail.com
Instructor email: bdolmail@gmail.com

Contents

1 Introduction

This lecture will cover setting up the development environment that will be used in future lessons for the Intro to 2D Rigid Body Physics (I2RBP). We will cover installing Python and the pygame module and go over how to display a window that will be able to show our graphics. The main focus of this class will be developing the simulation behind the graphics, but we will be using the framework we build in this lesson to test our engine.

2 Suggested reading

Although directions are provided here, students should go over the directions on how to install Python and pygame for their respective platforms at the official websites.

- http://www.python.org/
- http://www.pygame.org/

3 Installing Python and pygame

I have only tested this procedure myself on Windows and Macintosh. This should cover the majority of users. For those of you who are using some flavor of Linux, I've only summarized the directions on each project's homepage because I currently don't have access to a machine with Linux.

3.1 Windows

If Python is not installed, download the Windows installer. The installer binary (without source) is available here:

- http://www.python.org/download/releases/2.6.5

Although pygame recommends using Python 2.5.4, for consistency I am using Python 2.6.x across all platforms.

Next, download the pygame binary installer from here:

- http://www.pygame.org/download.shtml

Install both in the same directory. If you download the binary installers, the installations are simply point and click. After installing both Python and pygame, move to the section on coding the display window.

3.2 Mac OS X

The installation of Python and pygame on OS X is fairly straightforward. Even though you may already have a system version of Python 2.6, pygame will not install correctly using the default system version. Therefore, it is easiest to download and install Python 2.6.5 from here:

- http://www.python.org/download/releases/2.6.5/

Install the Python package from the disk image. Then download pygame 1.9.1 for Python 2.6 from this page:

- http://www.pygame.org/download.shtml

Install pygame from the archive, and move to the section on coding the display window.

3.3 Linux

If you don't already have Python and pygame installed on your Linux machine, I will briefly go through the steps needed to compile it from source. You may be able to install the binaries from a repository (for instance, by using apt). However, **Python and pygame are usually already installed on most Linux distributions**.

To build Python from source, first download the latest v2.6 Unix tar ball from:

- http://www.python.org/download/releases/2.6.5/

Next, run the `customize.sh` shell script to create the Makefile. See the README in the root directory of the Python source for configuration options. Next run `make` using the generated Makefile (you will need GNU Make for this step). Finally, run `make install` to install the Python interpreter and its modules.

Finally, we need to build and install the pygame module. Download the Unix format tar ball from:

- http://www.pygame.org/download.shtml

Unzip the directory and run `setup.py`. You can run `python setup.py help` for configuration options. Running `setup.py` will build and install pygame in the default location.

Finally, move to the section on coding the display window.

4 Coding the display window

Now that we have Python and the pygame module installed, we will code our display window that will display our physical objects. First create a directory where you are going to be keeping your source code. For instance, I simply created a `src` directory where I will be adding all my classes. In that directory, create a file called `rbp.py`. Open this file in your favorite text editor.

The first thing we have to do is import the pygame module so we can create a display window. In Python, it is easy to include external modules. Add this line to the top of your `rbp.py` file:

```
import pygame
```

This will import all the pygame functions that will be used. Next, we will define some constants. The `SPEED` constant defines the number of ticks per second that our simulation will run at. The rest of the constants are self-explanatory:

```
SPEED = 100              # The speed at which our simulation is run
SCREEN_WIDTH = 640       # The dimensions of the display window
SCREEN_HEIGHT = 480
C_BLACK = 0, 0, 0
```

Next, we actually initialize pygame and setup the display window:

```
pygame.init()
screen = pygame.display.set_mode( (SCREEN_WIDTH, SCREEN_HEIGHT) )
pygame.display.set_caption('I2RBP')
```

The next two variables are used in controlling the simulation updates. The `Clock()` pygame function is an object that provides a constant framerate.

```
clock = pygame.time.Clock()
```

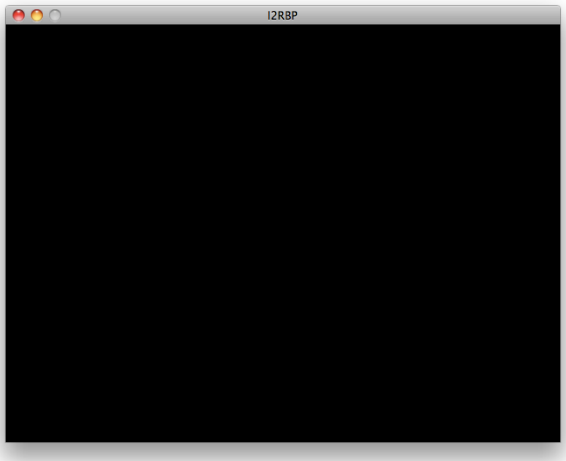
```
    running = 1
```

Finally, we add the drawing loop to our Python file:

```
    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = 0

        screen.fill(C_BLACK)

        # Insert drawing code here

        pygame.display.flip()
        clock.tick(SPEED)
```

Right now we don't actually draw anything (except a black window). In the future, we will be accessing a data structure that stores all the physical objects. Each object will have its own `draw()` method, so we only need to iterate over the structure and call `draw()` on each object.

That is all the code for this module. If we run `python rbp.py`, you should see the following window (the appearance will vary by platform).



The full source code can be found in the Code section.

5 Conclusion

This module was fairly explicit in exactly how to code each portion of the lesson. In future lectures, the focus will be on theory as opposed to coding specifics. All source code will be provided, but may not be explicitly covered in the lesson. I recommend that you attempt to code some of the ideas presented in each lesson yourself before you look at my source code.

6 Keyword Definitions

Python

A general-purpose high-level programming language whose design philosophy emphasizes code readability[1]. Python is used for all code examples in I2RBP.

pygame

A set of Python modules designed for writing games[2]. In I2RBP, it is used as an easy interface for displaying and updating a graphics window.

7 Code

`rbp.py`

```
import pygame

SPEED = 100              # The speed at which our simulation is run
SCREEN_WIDTH = 640       # The dimensions of the display window
SCREEN_HEIGHT = 480
C_BLACK = 0, 0, 0

pygame.init()
screen = pygame.display.set_mode( (SCREEN_WIDTH, SCREEN_HEIGHT) )
pygame.display.set_caption('I2RBP')

clock = pygame.time.Clock()

running = 1
```

```
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = 0

    screen.fill(C_BLACK)

    # Insert drawing code here

    pygame.display.flip()
    clock.tick(SPEED)
```

8 References

1. http://en.wikipedia.org/wiki/Python_(programming_language)
2. http://www.pygame.org/wiki/about