Writing fun games in python

and learning object-oriented programming during the process

Guillermo Aguilar

October 14th, 2014

Graphics on python

PyOpenGL

Game engines

and hundreds more.. ▸ Link

Let's program!

Step 0: Clone the repository

Clone the repository

git clone https://github.com/guillermoaguilar/pyglet_tutorial.git

Step 1: Establishing a window

- Test that pyglet can indeed start a window. For that, run the file *game1.py*
- Examine the code
  - You have two class definitions: **SpaceGame** and **Spaceship**. What do these represent?
  - How do the program execution work in pyglet?
  - How do the FPS text is created and drawn?

Step 2: Draw the spaceship

- You need to first load the image of the spaceship. Use *pyglet.image.load*.
- Then, instantiate an **Spaceship** object passing the image data as an argument. You can also pass variables x and y. Write these two commands in the initialization method of the object **SpaceGame**.
- Add a call for drawing the spaceship in the draw method of **SpaceGame**.
- Run it !!

Step 3: Moving the spaceship

- Make the spaceship location change according to the mouse position.
- For that, use the event handler *on_mouse_motion* to update the position of the spaceship object.
- Lock the spaceship at *y=50*, so that it can only move left-right with the mouse.

Solution until this point: *game2.py*

Step 4: Adding the alien invaders

- Similarly to the spaceship, first you need to load the image of an alien.
- Create a new class **Alien**, copying the definition from the **Spaceship** class.
- Similarly, instantiate a new **Alien** object in the game, and draw it.
- Add a velocity variable to the **Alien**, and add an update() method that will be called at every frame step. Inside the update method, update the position of the alien, so that moves downwards.

At this point (*game3.py*), you have a spaceship and an alien in your game. But this is pretty boring, isn't it? We need to add some interaction and movement! We need more aliens!

Step 5: Adding many aliens

- Initialize an empty list aliens in the *(__init__)* method of **SpaceGame**
- Create a function *create_alien()* that appends to the aliens list one alien, at random horizontal positions.
- Add a call to *clock.schedule_interval*, calling the *create_alien()* function.
- On the update method, update every alien from the list.
- On the draw method, draw every alien.
- And make sure that the alien is cleared when approaches the bottom of the screen. For that, add a *dead* variable to the **Alien** object, and at every update check if the alien is dead. If so, remove it from the list.
- To make it less boring, set the horizontal velocity random for each alien.

Solution until this point: *game4.py*

Step 6: Adding bullets, and collision detection

Finally, we need to add the bullets that the spaceship will shoot. And we'll check that if an alien receives a bullet, it's killed!

- Create a **Bullet** class definition, and add the bullet image similarly as before. Also create an empty list of bullets.
- Bullets are now created every time the user clicks. Add this on the handle event *on_mouse_press*.
- Make sure bullets are created where the spaceship is located.
- Write a distance function outside the class definition. Use this function inside the update method of the game, to check and detect collisions between aliens and bullets. If detected, set *dead=True* to both. And also use it to detect a collision between an alien and the spaceship. If detected, terminate the game.
- Finally, add a text on the upper right corner that keeps track of the number of aliens killed. Add an attribute to the **Spaceship** object that keeps track of that.

Final version! : *game5.py*

Optional modifications

- Modify the alien class so that after x steps of motion, changes its x velocity randomly (as in *game.py*)
- Instead of having a list of objects for the bullets and aliens, use the capabilities of Batch Sprites on pyglet. This should improve performance.
- Add a collision image and a sound every time a collision is detected. Add sound for the shooting!
- Whatever you can think of !

Have fun coding!!