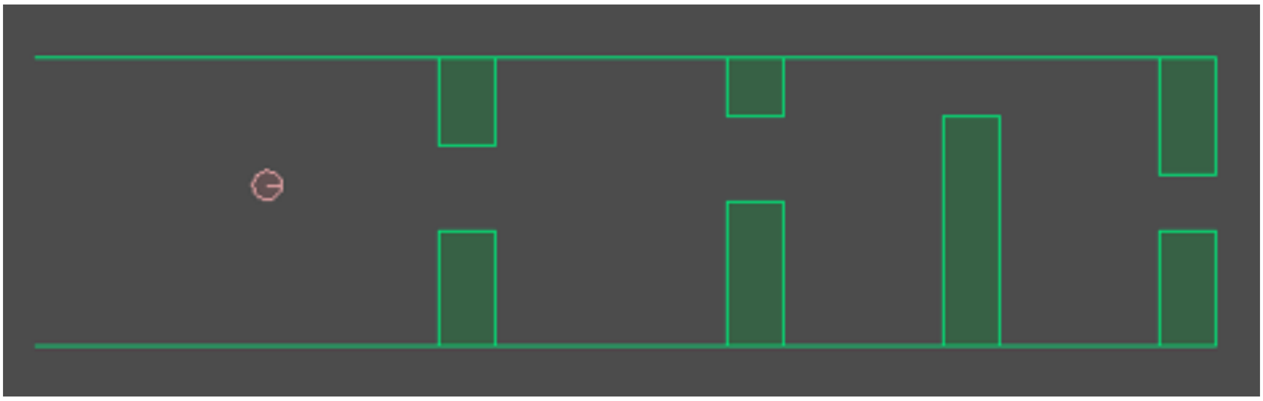
Tuesday 4 October 2016

Physics exercises (3 pages)

Exercise 1 - Create a Box2D testbed Test

- Create a testbed test scene, that looks like this:



Hints:

- Here's a guide on how to make a testbed test, and other Box2D tutorials
  - http://www.iforce2d.net/b2dtut/making-a-test/
  - Make sure the above link didn't take you to the first page, but if it did, just use left sidebar for navigation.
- You might want to take a look at the other tests, in the testbed, specifically:
  - Tiles.h
  - SphereStack.h
  - CharacterCollision.h
- Sizes and Types:
  - Make Ground and Ceiling 20 meters long (x axis), and 10m apart (on the y axis).
  - Pillar #1: Create as a b2PolygonShape, and set:
    - shape.SetAsBox(1.0f, 2.0f, b2Vec2(-5.0f, 2.0f), 0.0f);
    - Make sure to fixture it to the ground (See above tests, for shape creation examples)
  - Pillar #1 top: Create as a b2PolygonShape, and set:
    - shape.SetAsBox(1.0f, 1.5f, b2Vec2(-5.0f, 8.5f), 0.0f);
  - Create the rest of the pillars, to roughly match the above picture.

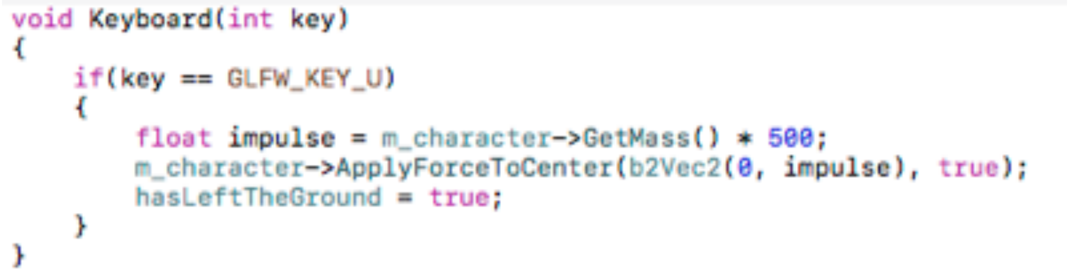
- Create the character as a b2CircleShape.
  - at the position (-19m, 1m)
  - make sure it's dynamic
  - give it a radius of 0.5m
  - a density of 20
  - a friction of 1 (not really important for our case, as we'll be in the air mostly)
  - and make sure to SetFixedRotation(true) on it

Exercise 2 - Add movement to the test

- If you hadn't already noticed, this test looks an awful lot like a flappy bird style game.
- So let's make the player move, in the following manner:

1. First, while no buttons have been pressed, only gravity should apply (which is automatic in the testbed).
2. Once the user presses 'U button' [Worth noting that the SPACE key is used by the testbed to create a "Bomb"], apply an upwards impulse to the player.

- Use the following keyboard handling code, for this.



```
void Keyboard(int key)
{
    if(key == GLFW_KEY_U)
    {
        float impulse = m_character->GetMass() * 500;
        m_character->ApplyForceToCenter(b2Vec2(0, impulse), true);
        hasLeftTheGround = true;
    }
}
```

-

3. Now that the player is no longer touching any colliders, add a constant velocity to the right.

- Set the velocity to 3 m/s on the axis
- use b2Vec2 v = m_character->GetLinearVelocity(); to get the current velocity
- and once you've changed the x axis to 3 m/s, use
- m_character->SetLinearVelocity(v); to set it again.

4. If the player hits either ceiling, floor or obstacles, restart the test.

- you can implement the void BeginContact(b2Contact* contact) function for this purpose.

  - Do note that restarting means setting the player position, velocity and hasLeftTheGround, back to their default values, and that this cannot be done while the step function is executing.
- Tweak impulse, velocity and density of the player until you can complete your test/game.

Exercise 3 (Mandatory hand-in) - Combine your test and your engine, to create a simple flappy bird game, in your own engine.

- Using Simple Render Engine and the b2debugDraw (Box2D) class, draw the physics world directly in your window, making it look roughly like the testbed implementation.
- Using the world you built for the testbed test, implement the flappy bird game in your own engine.
- Again make sure the game is playable, and you can complete it yourself.
- Hints:
  - Use the provided CMake project that includes both SRE and Box2D, as a reference for setting up your own game engine CMake projects.
  - The test project comes with a simple example of keyboard ‘Space’ key calling a function, you may use this as a starting point for the exercise.
  - Box2D allows you to implement a debug draw class, to allow it to draw the physics world, yo can see how to do that here:
    - http://www.iforce2d.net/b2dtut/debug-draw/
    - AND SimpleRenderEngine allows you to draw lines like this:
      - Debug::drawLine(vec3(0,0,0), vec3(width,height,0));

Exercise 4 (optional) - Combine your game with your sprite rendering.

- Using the earlier sprite rendering exercise result, and Thursdays scene graph exercise, create a sprite rendered version of your flappy bird game.