**Lab 7 – Gravity & Advanced Collision**

**Step 1 – Platforms**

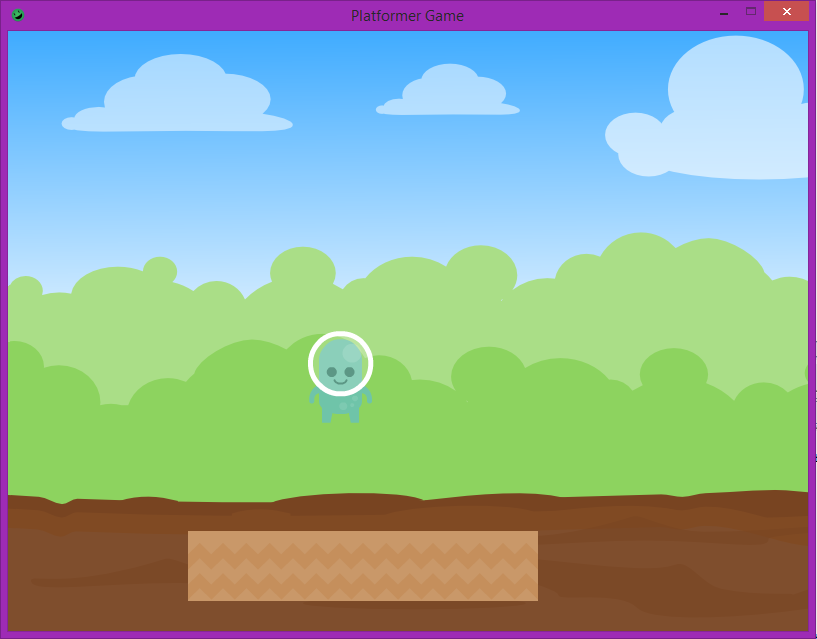
In order to make our platformer game, we need to actually have some platforms for our **Player** to jump and land on. Let’s create a **Block** class and extends from **Sprite**. This class will be very small and similar to how we did our **Player** class in the last lab. Create a constructor that takes 2 int parameters which are **x** and **y**. In the constructor, call the superclass’s constructor and pass those same **x** and **y** parameters, plus the image path “block.png”.

The “block.png” picture is 70 pixels wide by 70 pixels high. What we think of as a platform in a game isn’t just one individual platform, but actually multiple block pieces next to each other. For example, if we wanted to create a 5 **Block**, horizontal platform, we want to positon 5 **Blocks** next to each other that are 70 pixels apart.

Go back to the **PlatformerGame** class and let’s try creating a platform. Do the following:

* Add the following import to your class: import java.util.\*;
* Create a private **ArrayList<Block>** variable called **platform**
* **init()**
  + Initialize our **platform** variable
  + Add **5** **Block** variables to our **platform** list. The first **Block** should start at (180, 500) with each subsequent block being 70 pixels after the previous
* **render()** – Loop through the **platform** list and call the **render()** method on each **Block**

Compile and run you code, you should now see a platform of blocks below where your **Player** is!



**Step 2 – Gravity**

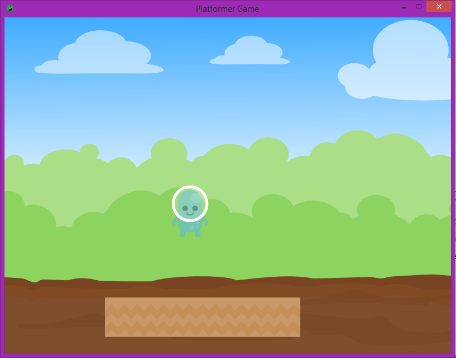
We can see the platform now, but we need our **Player** to actually be able to land on the platform. To do this, we to add gravity to our **Player** so that they can fall on to the platform. Let’s talk about how gravity works, as far as games are concerned. Every frame, a downward force is applied to an object, which always starts at 0 and is increased by 1 every frame. If the object collides with something, the downward force is set back to 0 and the object is shifted upward so it no longer collides with it. If there is no collision, then the downward force is maintained and will increase next frame. Let’s look at an example:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Velocity = 0  Velocity + 1 = 1  No collision  Go to next frame | Velocity = 1  Velocity + 1 = 2  No collision  Go to next frame | Velocity = 2  Velocity + 1 = 3  Collision  Go to adjust position | **Adjust Position:**  Get distance between  Adjust player  Velocity = 0 |

Let’s go to our **Player** class and add some gravity to it. Do the following inside the class:

* Add a private int variable called **yVelocity** and set its initial value to 0
* **update()** – At the end of our method
  + We need to increase **yVelocity** by 1
  + Move our **Rectangle** by the **yVelocity** amount

Compile the **Player** class, then go back to the **PlatformerGame** class and run your game. You should now see our **Player** start the fall...right off the edge of the screen.

Even though we have a platform, we’re not checking to see if we have collided with it. Let’s go to our **Sprite** class and add this functionality. Create a new method called **intersect( Sprite other )** which returns **true** if our **Rectangle** intersects with the **other** **Sprite’s** **Rectangle**. Compile the **Sprite** class and let’s make our way back to the **Player** class.

Now that we can check for collisions, we have to stop our downward momentum. Let’s create a method called **adjustForCollisionBelow( Sprite other )**, which will be in charge of fixing our **Player’s** position when we collide with something below us. Let’s quickly walk through what we need to do here.

|  |  |  |
| --- | --- | --- |
|  | 16  20 | 16 |
| If Collision occurs  **- adjustForCollisonBelow()** | Get bottom position of **Player** (20)  Get top position of platform (16)  Subtract top from bottom (16 – 20 = -4) | If difference < 0  - Move **Player** by difference (-4)  - Velocity = 0 |

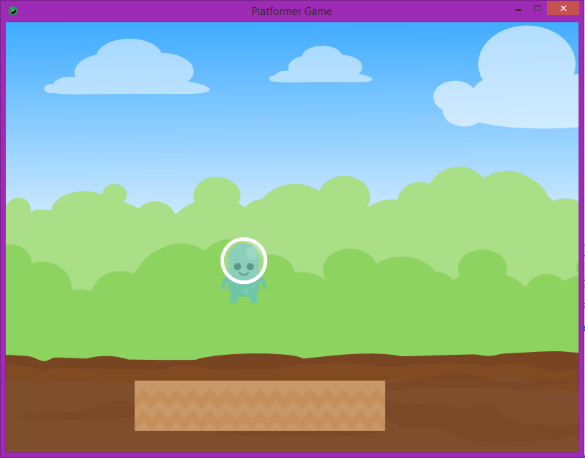
Let’s summarize what we need to do in **adjustForCollisionBelow( Sprite other )**:

* Subtract the **top** position of the **other Sprite** from the **bottom** position of **Player**
* If the difference < 0 (**Player**’s bottom is below the **Sprite’s** top)
  + Move the **Player** by the difference
  + Set the **Player’s** velocity to 0

Once this is done, compile your **Player** class and let’s head back to the **PlatformerGame** class. Now that we can both check for collisions and adjust our **Player** accordingly, let’s actually add the code to before both of these. Go to your **update()** method and add the following to the **END** of the method:

* Loop through every **block** in the **platform** list
  + If the **Player** **intersects** with the current **Block**
    - Call the **adjustForCollisionBelow( Sprite other )** method

Compile and run your code, you should now see your **Player** fall, but then stop when he gets to the platform below him! For extra knowledge, try moving off either edge of the platform. Since the **Player** is not colliding with anything below him anymore, it falls off the edge of the screen.

**Step 3 – Jumping**

Since we have gravity, we also have to allow the player to fight back against gravity by jumping. Since our **yVelocity** controls the force that’s applied to our **Player**, we simply need to change its value to apply a new force to it. We apply a **positive** force to our **Player** to make them fall, so we just need to apply a **negative** force to our **Player** to make them jump. Go to our **Player** class and do the following:

* **update()**
  + **IMMEDIATELY BEFORE** we add 1 to our **yVelocity**
    - Check if the **space** key has been pressed
      * Set our **yVelocity** equal to -20

Compile your **Player** class and then run your **PlatformerGame** class. If you press down the **space** key, you should see the **Player** jump!...and keep going up. As long as we are pressing the **space** key, we’ll keep setting our **yVelocity** equal to -20, which mean we’ll keep moving upward really fast. What we really want to say it: “Only let us jump if we’ve **touched the ground**”. If we have collided with the ground, then we want to keep track of that. However, the next time we go into **update()**, we want to assume we are not still touching the ground because the **Player** could have moved again. Go back to your **Player** class and make the following changes:

* Add a private boolean variable called **isGrounded** and set it to false
* **update()**
  + Change the if statement we just made to ALSO check if **isGrounded** is true
  + At the end of the method, set **isGrounded** to false
* **adjustForCollisionBelow( Sprite other )**
  + Inside of the if statement, set **isGrounded** to true

Compile your **Player** class then run your **PlatformerGame** class. Now when you pressed the **space** key, your **Player** should jump, but only when he is touching the ground!