This lesson follows the Construct 2 platformer tutorial found [here](https://www.scirra.com/tutorials/253/how-to-make-a-platform-game). This whole tutorial covers creating a platformer from beginning to end with no knowledge of Construct 2.

**Objective:**

At the end of this lesson, the students will be able change between animations, detect input, and create enemies with basic AI behavior.

**Input**

Right now, the Player sprite can move around the Layout, but it does not turn when moving left or right. We need to tell the Player Sprite to flip when the corresponding arrow is pressed.

1. **Adding Keyboard Detection**
   1. If you are not in the Layout, switch now.
   2. Double-click on an empty part of the Layout.
   3. Double-click the Keyboard object from the Input section of the Insert New Object pane. Now you can create events for keyboard input.
2. **Create Keyboard Events**
   1. Switch to the Event Sheet.
   2. Double-click an empty spot in the Event Sheet or click the “Add Event” button.
   3. Double-click the Keyboard object.
   4. Double-click the “On key pressed” event found in the Keyboard section.
   5. Click the <click to choose> button.
   6. From here just press the key you would like to detect input from. Press the Left Arrow now.
   7. Click “OK” then click “Done”. This closes the Add Event pane.
   8. Click the “Add action” button.
   9. Double-click the Player.
   10. Double-click the “Set mirrored” option.
   11. Leave the state on “Mirrored” and click done.
   12. Add another event for pressing the Right Arrow.
   13. Click the “Add Action” button and follow the above steps but set the “Mirrored” stat to “Not Mirrored”. Now when we press the left and right arrows, the player will flip to point in the correct direction.

**Adding More Animations**

1. **Adding a Walk Animation**
   1. Double-click the Player sprite.
   2. The Image Editor pane will pop up.
   3. Right-click in the Animations pane and select “Add animation”
   4. Rename the Animation to “Walk”
   5. Just like the tiles, right-click on the Animation frames pane and highlight “Import frames” then select “Import sprite strip...”.
   6. Use the provided animation of the player walking.
   7. Construct 2 will guess the number of frames since our image is not square. The horizontal frames should be 4 and the vertical frames should be 1. If not change the values to match.
   8. Delete the first blank frame.
2. **Adding a Jump Animation**
   1. Right-click in the Animations pane and select “Add animation”
   2. Rename the Animation to “Jump”
   3. Just like the tiles, right-click on the Animation frames pane and highlight “Import frames” then select “Import sprite strip...”.
   4. Use the provided animation of the player jumping.
   5. Construct 2 will guess the number of frames since our image is not square. The horizontal frames should be 3 and the vertical frames should be 1. If not change the values to match.
   6. Delete the first blank frame.
3. **Loop the Animation**
   1. To loop an animation set the “Loop” property to Yes. This will make the animation run over and over again. Set the Walk animation to loop.

**Setting up Animation Events**

Now that we have multiple animations, we need to tell our player when to switch these animations. We do that with events.

1. **Switch to Walk Animation**
   1. Switch to the Event Sheet.
   2. Double-click the Event Sheet or select the “Add Event” button.
   3. Double-click the PlayerBox.
   4. Double-click the “On Moved” button in the Platform: Animation Triggers section.
   5. Double-click the “Add action” button next to the new event.
   6. Double-click the Player object.
   7. Double-click the “Set animation” button in the Animations Section.
   8. Type “Walk” with quotations in the Animation field.
   9. Click Done. This event tells the Player object to switch to the Walk animation when it starts moving.
2. **Switch to Idle Animation**
   1. Double-click the Event Sheet or select the “Add Event” button.
   2. Double-click the PlayerBox.
   3. Double-click the “On Stopped” button in the Platform: Animation Triggers section.
   4. Double-click the “Add action” button next to the new event.
   5. Double-click the Player object.
   6. Double-click the “Set animation” button in the Animations Section.
   7. Type “Idle” with quotations in the Animation field.
   8. Click Done. This event tells the Player object to switch to the Idle animation when it stops moving.
3. **Switch to the Jump Animation**
   1. Double-click the Event Sheet or select the “Add Event” button.
   2. Double-click the PlayerBox.
   3. Double-click the “On Jump” button in the Platform: Animation Triggers section.
   4. Double-click the “Add action” button next to the new event.
   5. Double-click the Player object.
   6. Double-click the “Set animation” button in the Animations Section.
   7. Type “Jump” with quotations in the Animation field.
   8. Click Done. This event tells the Player object to switch to the Idle animation when it stops moving.
4. **Switch Animation After Landing**
   1. Double-click the Event Sheet or select the “Add Event” button.
   2. Double-click the PlayerBox.
   3. Double-click the “On Landed” button in the Platform: Animation Triggers section.
   4. Now we will us an If statement. This is a sub-event allows us to perform different actions based on the current conditions passed to the statement. In this case, we will play the walk animation when we land and we are moving. We will play the idle animation when we land and we are not moving.
   5. To add a sub-event. Make sure that the On Landed event is selected. This is done by clicking the event to the left of the green arrow. You can aim for a small vertical gray box.
   6. Right-click and Highlight “Add”, then select “Add sub-event”. Now treat that sub-event just like any other event.
   7. Double-click the PlayerBox.
   8. Double-click the “Is Moving” button in the Platform section.
   9. Double-click the “Add action” button next to the new event.
   10. Double-click the Player object.
   11. Double-click the “Set animation” button in the Animations Section.
   12. Type “Walk” with quotations in the Animation field.
   13. Click Done. This event tells the Player object to switch to the Walk animation when it lands and is moving.
   14. Right-click and Highlight “Add”, then select “Add sub-event”. Now treat that sub-event just like any other event.
   15. Double-click the PlayerBox.
   16. Double-click the “Is Moving” button in the Platform section.
   17. Now, we need to check if the PlayerBox is not moving. Right-click on this sub- event and click the “Invert” option. This will check for the opposite of the condition provided.
   18. Double-click the “Add action” button next to the new event.
   19. Double-click the Player object.
   20. Double-click the “Set animation” button in the Animations Section.
   21. Type “Idle” with quotations in the Animation field.
   22. Click Done. This event tells the Player object to switch to the Idle animation when it Lands and is not moving.

**Adding Enemies**

Enemies are vital to almost all games. They usually include Artificial Intelligence that controls their actions and can destroy the player in a number of different ways.

1. **Adding the Enemy Sprite**
   1. Add a new Sprite object in the Layout.
   2. Import the provided Soldier sprite strip.
   3. Delete the first empty frame of the animation.
   4. Set the animation to looping.
   5. Crop out the extra whitespace from each animation.
   6. Set the origin to the bottom center of each animation frame.
   7. Close the Image Editor.
   8. Place the Soldier on a platform.
2. **Adding Enemy Events**
   1. Switch to the Event Sheet and add a new Event.
   2. Double-click on the PlayerBox object.
   3. Double-click the “On collision with another object” Event.
   4. Click the <click to choose> button and then choose the Soldier sprite.
   5. Now we use sub-events to check if the Player is falling or runs into the Soldier
   6. Add a sub-event to the above event.
   7. Double-click the PlayerBox object.
   8. Double-click the “Is falling” option.
   9. We should also check to see if the PlayerBox is above the Soldier.
   10. Right-click the “Is falling” condition and select “Add another condition”.
   11. Double-click the PlayerBox object.
   12. Double-click the “Compare Y” option.
   13. Set Comparison to “<Less than”. Set Y co-ordinate to Enemy.Y.
3. **Destroy Enemy**
   1. Close the pane and select Add Action from this event.
   2. Double-click on the Soldier object.
   3. Choose the “Destroy” option.
4. **Give the Solder the Ability to Move**
   1. Switch to the Layout.
   2. Select the Soldier sprite.
   3. Add the Platform behavior to the Soldier.
   4. Set “Default controls” in the Properties pane to No. This prevents user input from moving the Soldier. THis also allows us to tell the Soldier when to move with Events.
5. **Create Edge Markers**
   1. We need an Edge Marker to tell the Soldier when to turn around so it doesn’t fall off the edge.
   2. Add a new sprite filled with a transparent color, just like the PlayerBox.
   3. Set Initial Invisibility to Invisible just like the PlayerBox.
   4. Copy this sprite and place one on the left edge of the platform that holds the Soldier and one on the right side of the same platform.
   5. We will need a variable to keep track of what direction the enemy is going. Variables store things like numbers and text. They can be used for all kinds of things in a game. Health, Score and in our case AI.
   6. In the Properties pane, under the Instance Variables section, click the “Instance variables” button. This will bring up a dialog box with all of the variables associated with this object.
   7. Click the “+” button to add a new variable.
   8. Set the name to “action”, Set the type to “Text”, Set the Initial value to “right”. Click “OK” to close.
6. **Create Soldier Move Events**
   1. Switch to the Event sheet and add a new Event
   2. Double-click on the Soldier object.
   3. Select the “Compare instance variable” in the Instance Variable section.
   4. Set Instance variable to “action”, set Comparison to “= Equal to”, set value to “right”. Make sure to use the quotations. Click “OK”
   5. Add an action to this new event.
   6. Double-click the Soldier object.
   7. Select the “Simulate control” option. Set Control to “Right”, click Done.
   8. Add another action to this event.
   9. Double-click the Soldier object.
   10. Select the Set mirrored option. Set State to Not Mirrored. Click OK.
   11. Select the previous Event. Remember to aim for the small vertical rectangle.
   12. Right-click the Event and highlight the Add option. Select “Else”
   13. Add an action to this new event.
   14. Double-click the Soldier object.
   15. Select the “Simulate control” option. Set Control to “Left”, click Done.
   16. Add another action to this event.
   17. Double-click the Soldier object.
   18. Select the Set mirrored option. Set State to Mirrored.
7. **Check for collision with the Edge Markers**
   1. Create a new Event.
   2. Double-click the Soldier object.
   3. Select the “On collision with another object” and select the Edge Marker object.
   4. Add a sub-event to this Event.
   5. Double-click the Soldier object.
   6. Select the “Compare instance variable” option. Set Instance variable to “action”, Set Comarison to “= Equal to”, Set Value to “right”, make sure to use quotations.
   7. Add an action to this sub-event.
   8. Double-click the Soldier object.
   9. Select the “Set value” option. Set the Instance variable to “action”, set the Value to “left”, make sure to use quotations. Click Done.
   10. Add an Else to this sub-event.
   11. Double-click the Soldier object.
   12. Select the “Set value” option. Set the Instance variable to “action”, set the Value to “right”, make sure to use quotations. Click Done.

Now, you should have a patrolling enemy! This enemy won’t kill the Player, but the player can get rid of the enemy. The Player also switches it’s animations when you jump, move and stop.