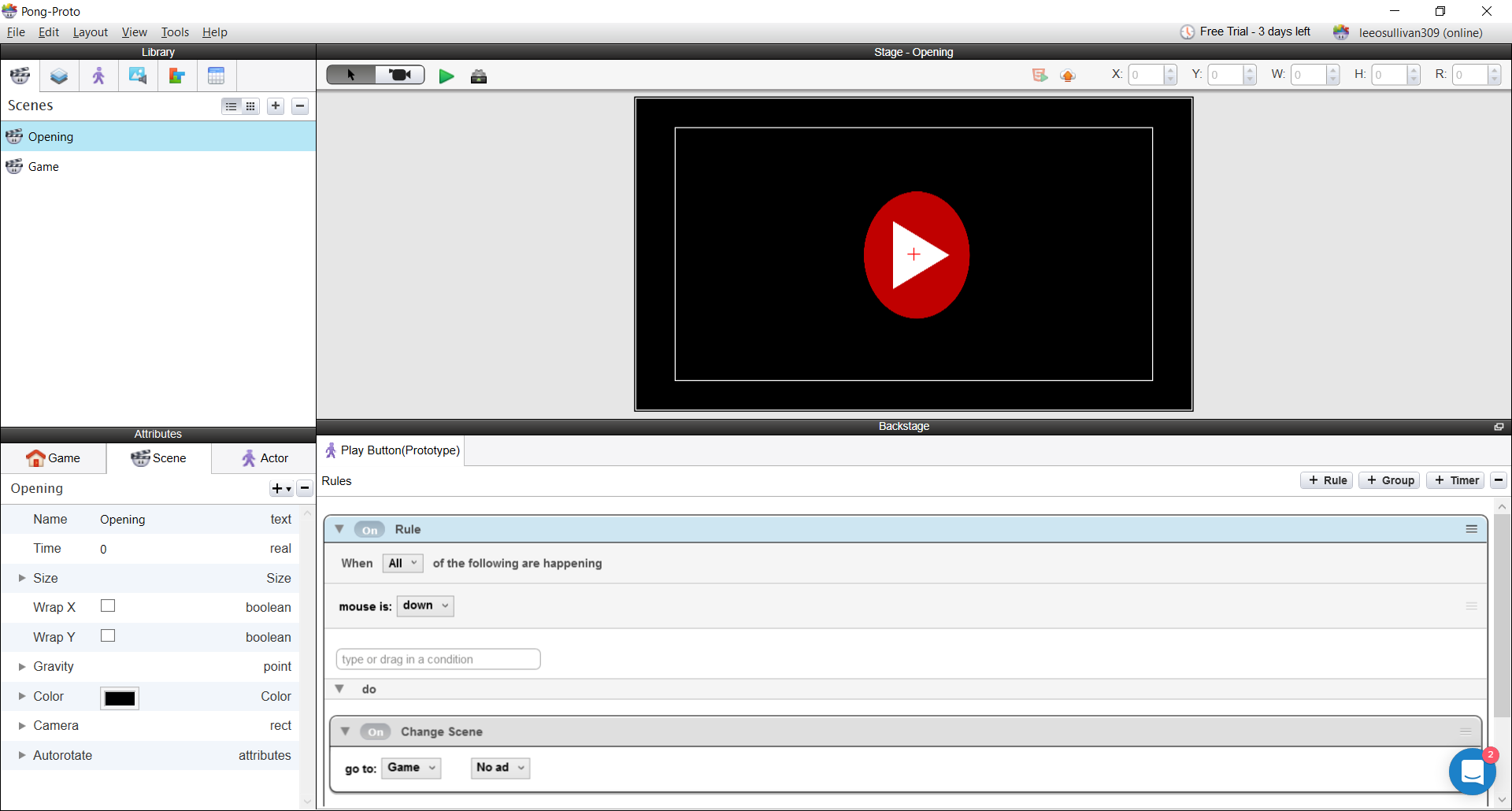
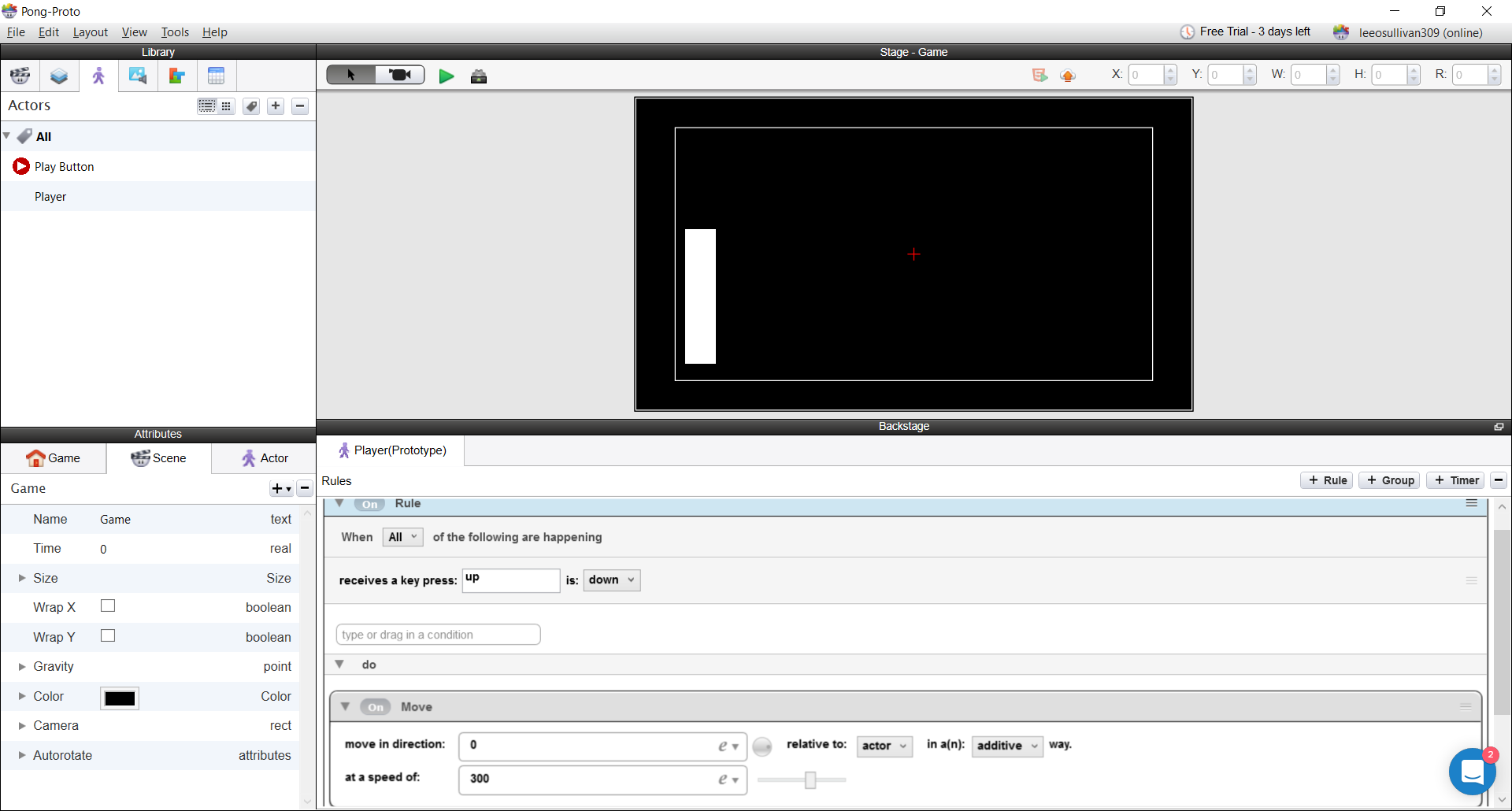
Rapid Applications Development

Assignment #1 – Pong Simulation

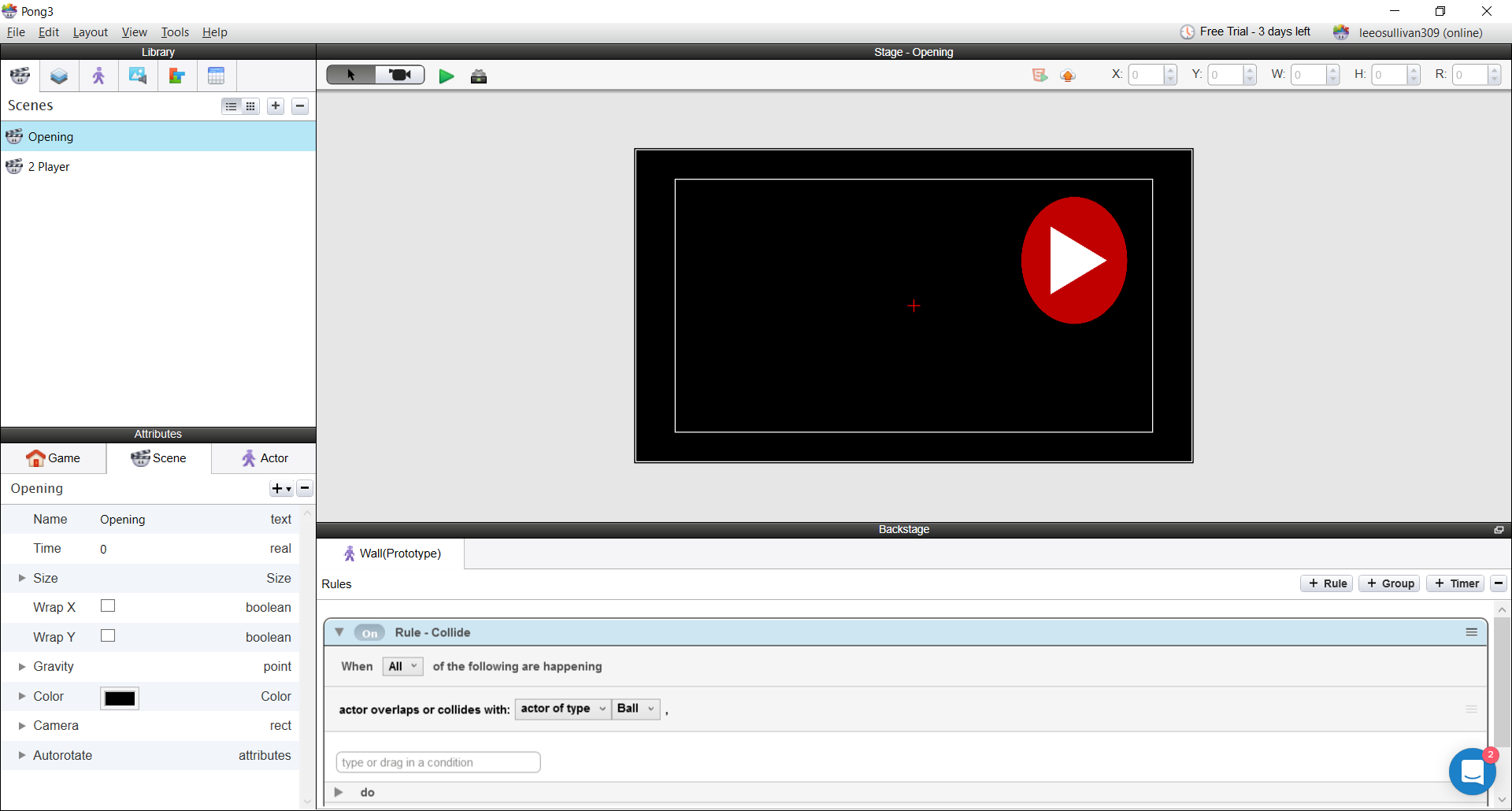
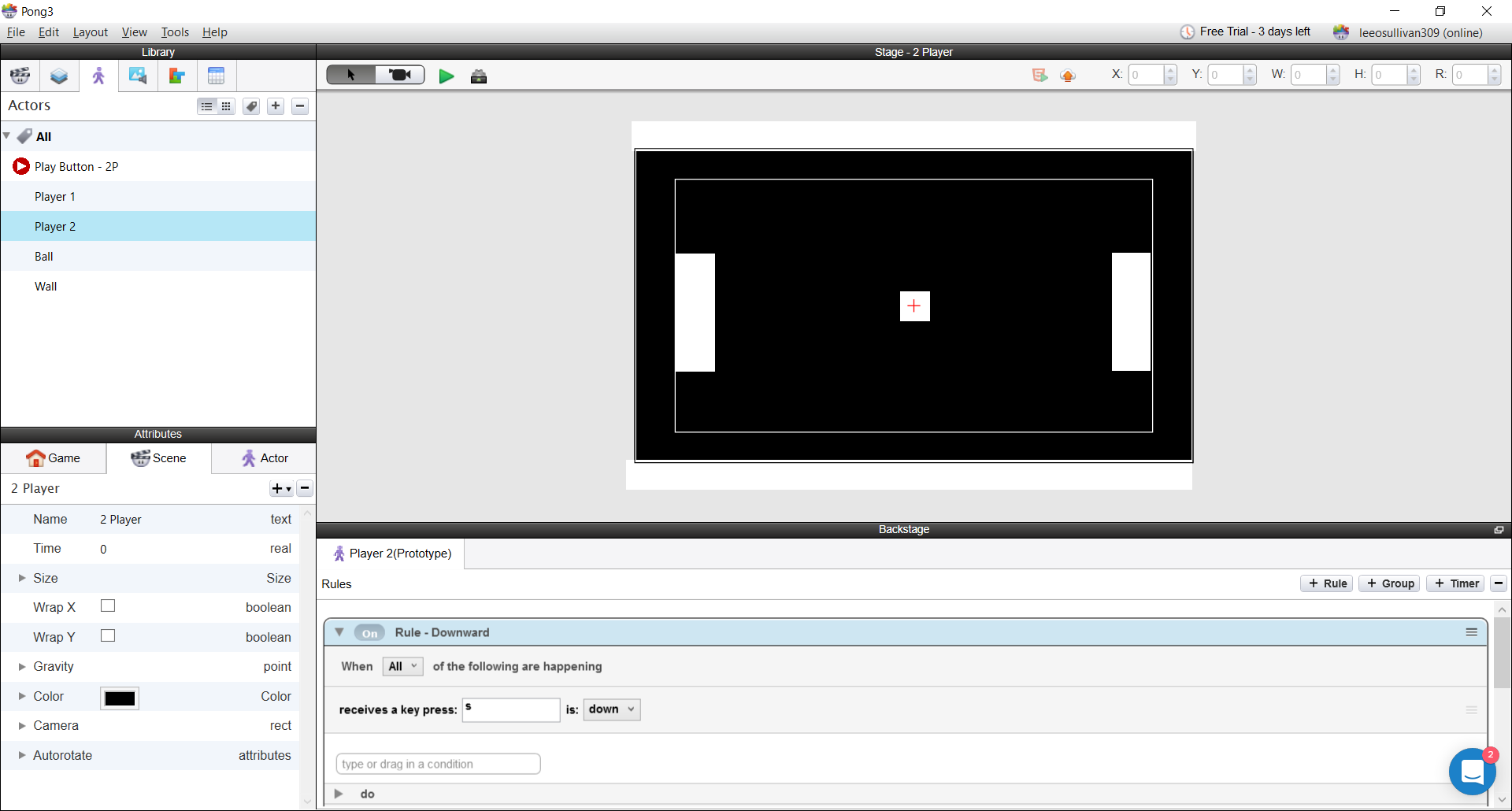
Lee O’Sullivan – T00195739

GitHub Repository - <https://github.com/Leeos2009/Rapid-Apps-Development-Assignment>

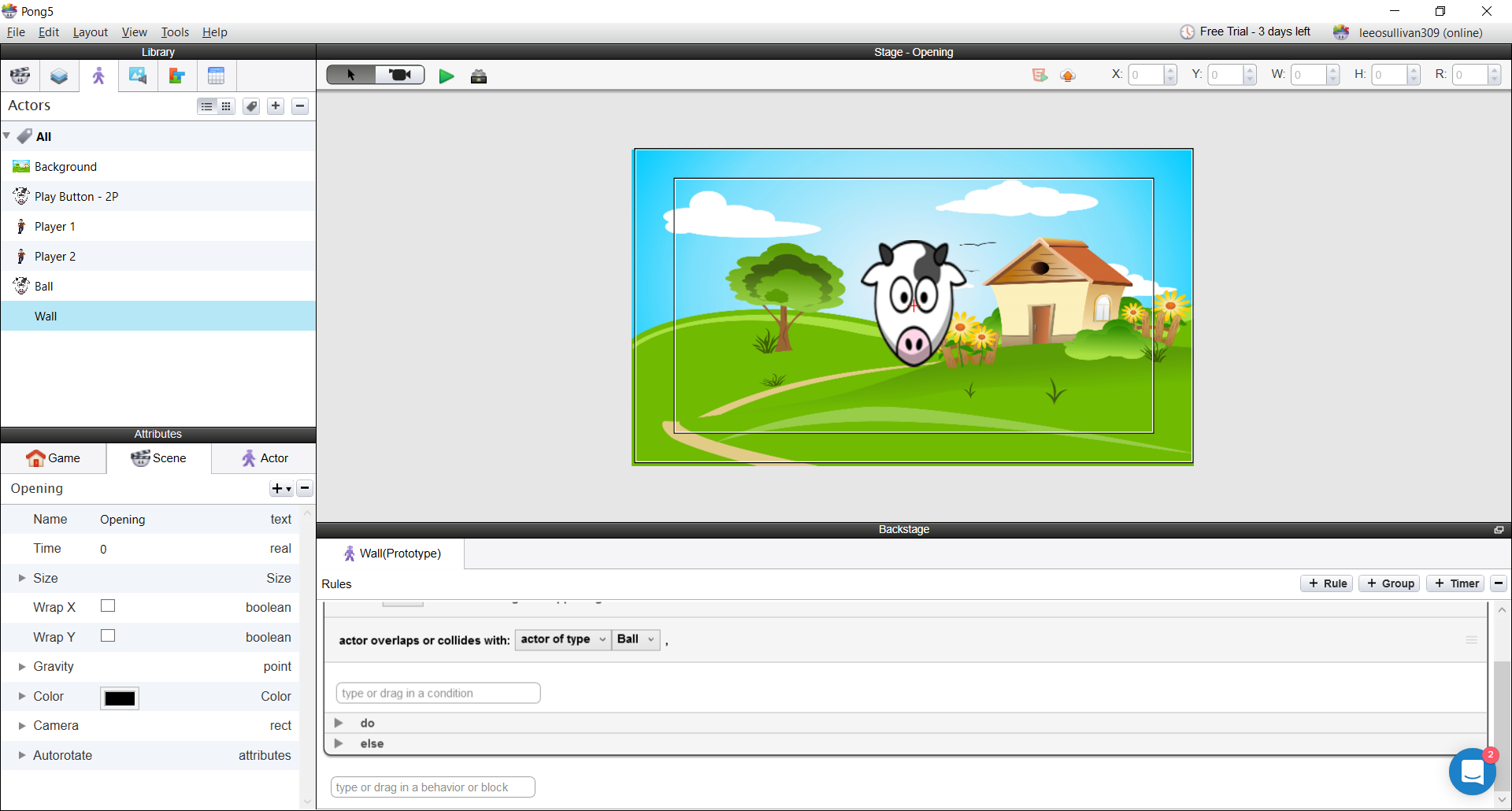
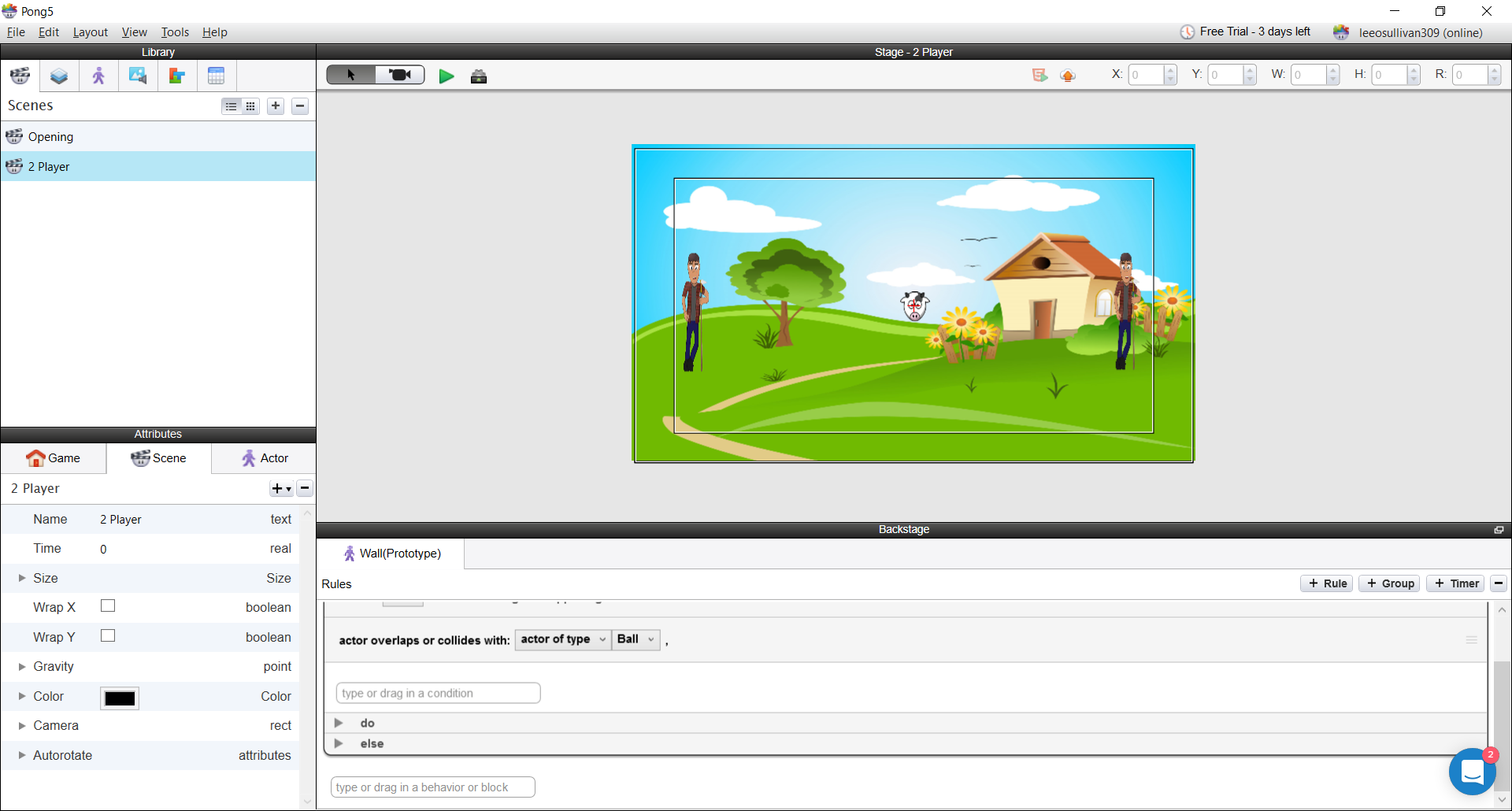
Version 1 -

To start this app, I made two scenes, as seen above. In the first scene I added an actor labelled “Play Button”. I added a rule which would change the scene when the screen was clicked. In the second scene, I added an actor labelled “Player”, to which I added one rule to allow the “Player” to move to the right.

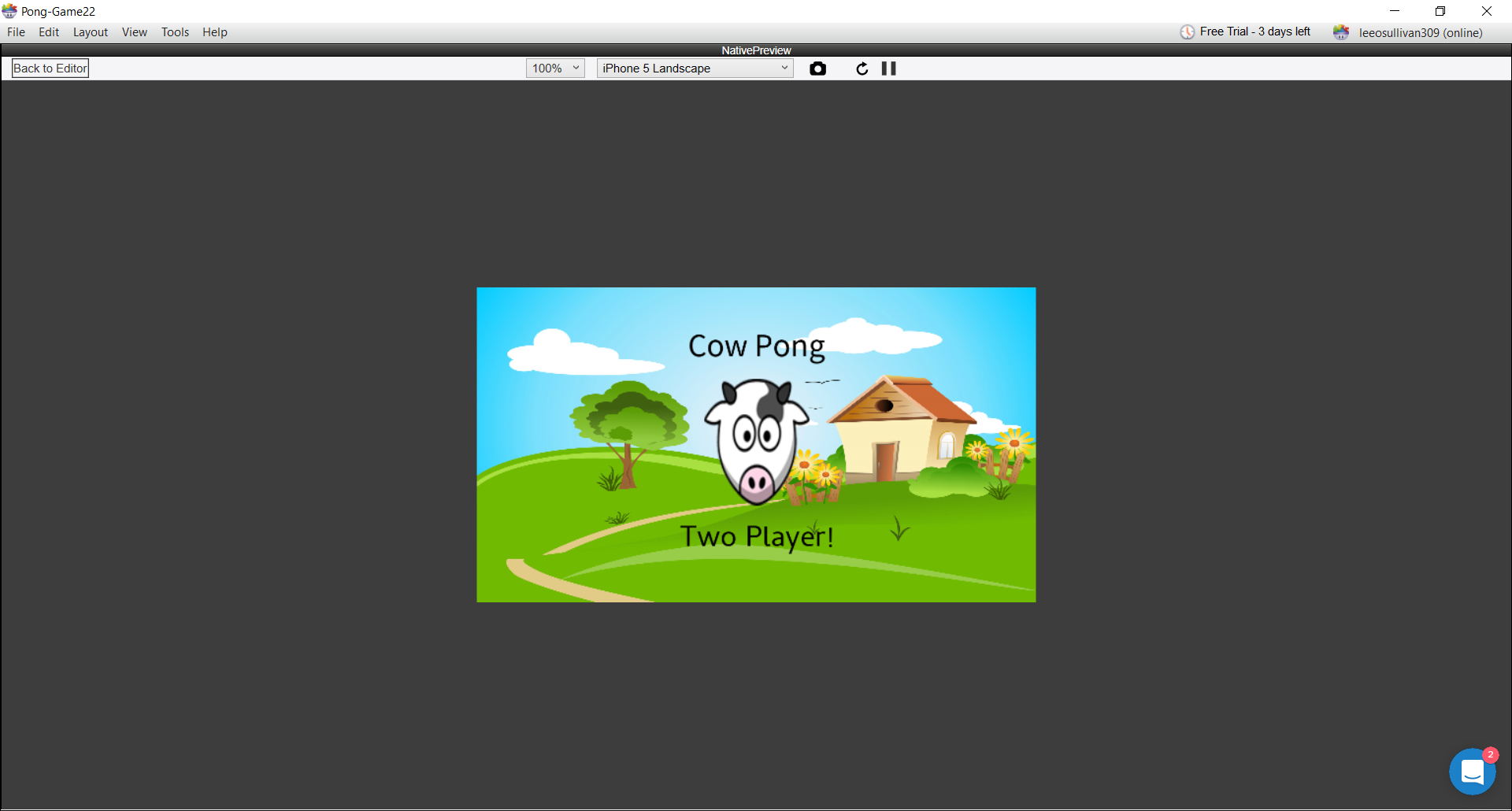
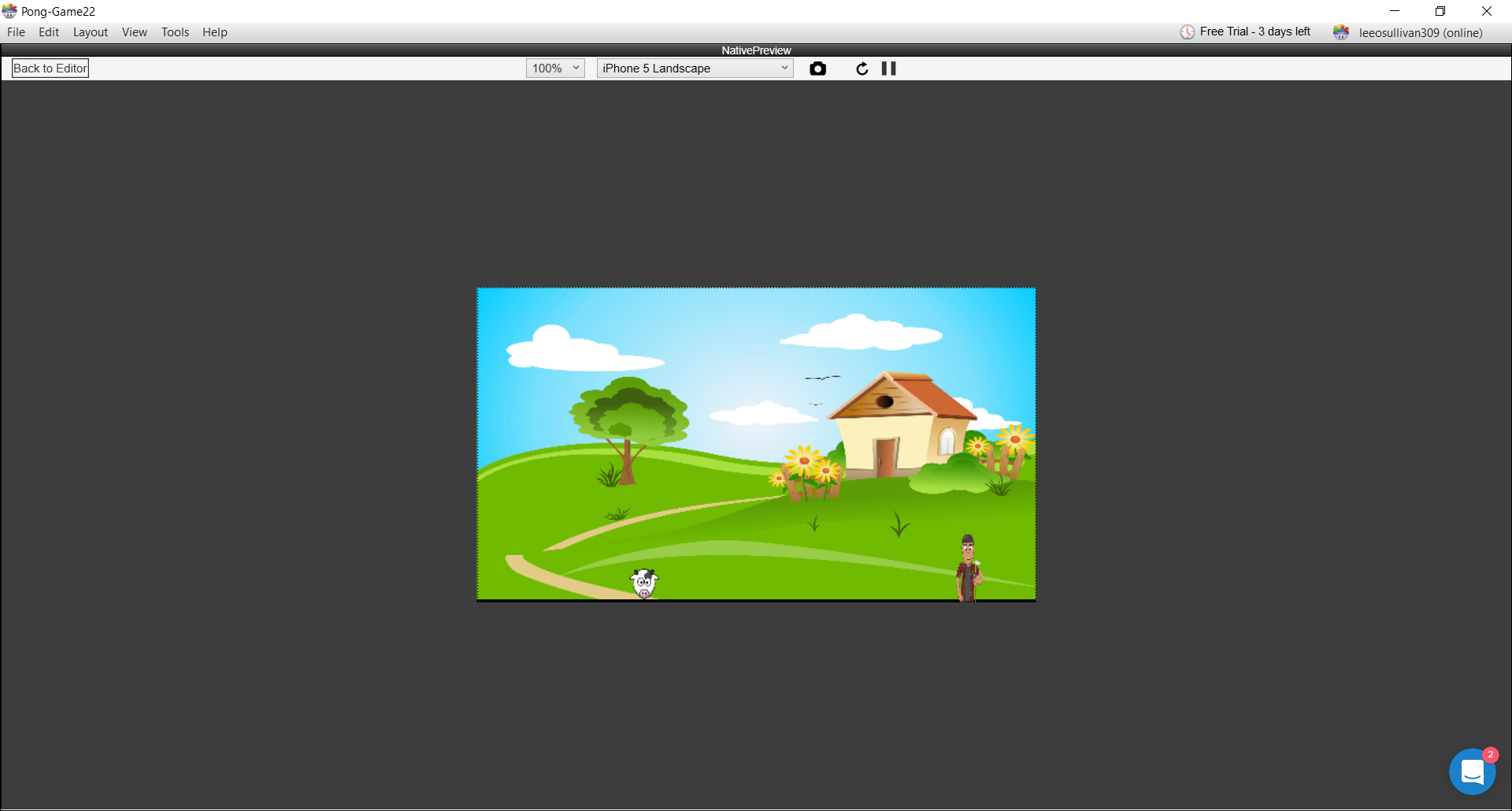
Version 2 –

To continue, I altered the rule for “Player 1” to move at 0 degrees and changed it to 90 degrees and I also added another rule which allowed the actor to move at down at 270 degrees, I also added another user-controlled character named “Player 2” with the same rules as “Player 1” so it could also move up and down with the use of the keyboard. Two more actors were added named “Ball” and “Wall”.

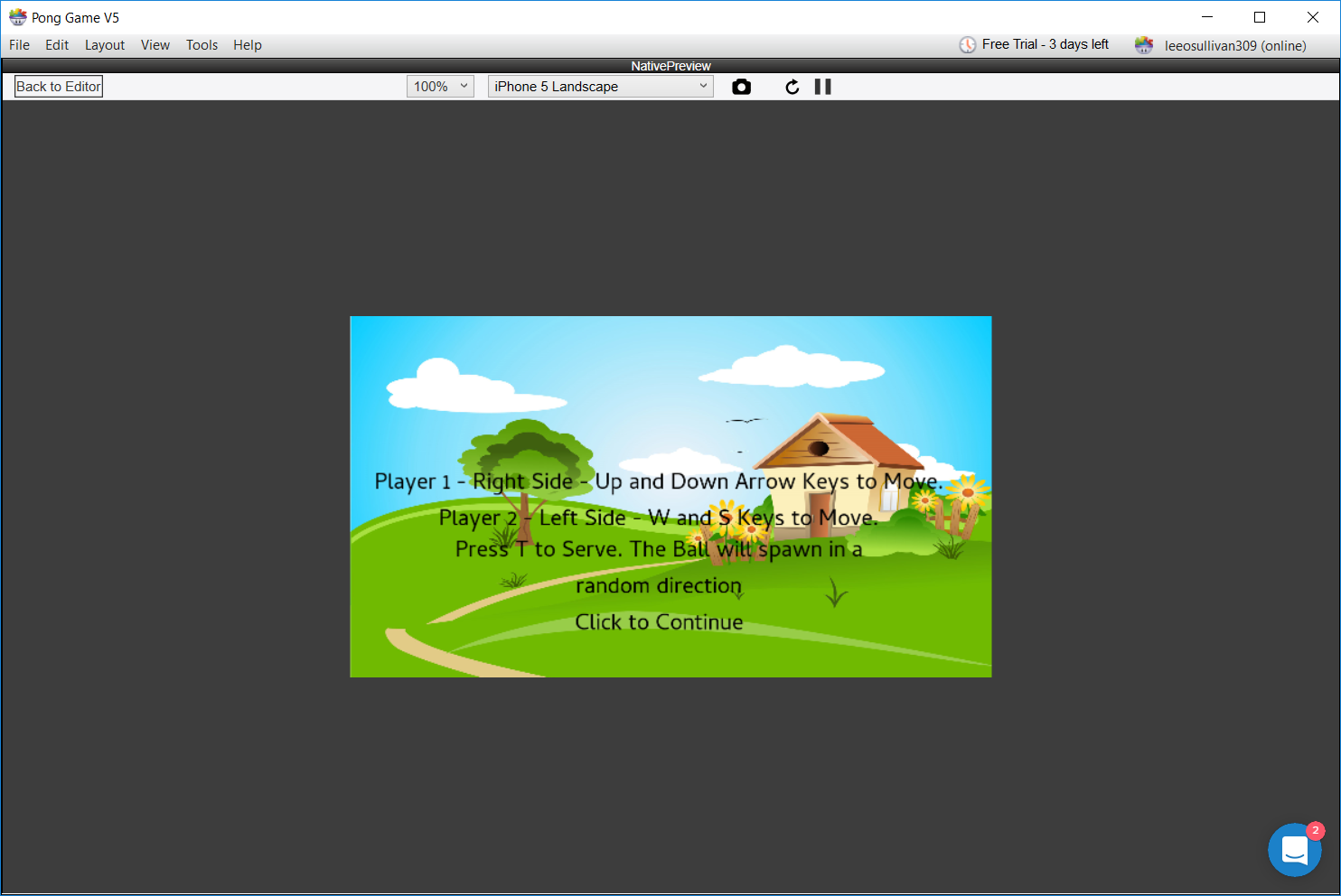
Version 3 –

A “Ball” actor was added and a change velocity rule was added to allow the ball to move freely at a random angle and at a speed of 320. I then added a collide rule to the “Player 1” and “Player 2” actor, so upon collision the ball would bounce off the characters. To stop the ball from bouncing off screen, I added an actor name “Wall” and placed it along the top and bottom of the screen and in the graphics section for the “Wall” actor I unchecked the visible box so the players wouldn’t see the wall. To keep the ball moving at a constant speed I added a bounce rule to the wall which caused the ball to bounce off the wall. In the opening screen, I added a display text rule which placed text above and below the starting button, which states the name of the game and how many players it requires.

Version 4 –

As the ball bounced off player 1 and 2, they would be pushed off the screen, so I added a constrain rule which confined the player actor at a specific X value while still allowing the actor to move up and down with the use of the keyboard. To calculate each player’s score, I added an integer for player 1 and 2 with a value set at 0. I also added a Boolean value to set the requirements for the ball to spawn and for the ball to be served into the screen at a random angle.



Version 5 –

To finish the game, I added the two goals and made them invisible through the graphics. I gave them a collide command, which would destroy the ball whenever they came in contact, I used a rule which spawns the ball through a key press and setting the “Serving” boolean value to true and used the spawn actor command to spawn the ball and using functions I was able to set the ball to spawn in a random direction. By making two actors and adding a display text command to each of them and inputting the “P1 Score” or “P2 Score” into the command it would display the player’s score on their own side. I then added an attribute which would increase the score by +1 for the player who scored in the opposite goal. I made a new actor which displays text when a player reaches 11 goals. I done this through an attribute command which stated when “P1 Score” was equal to 11 or “P2 Score” was equal to 11, a text message would appear stating the winner and how to reset the score and restart the game through the use of a rule and setting the attributes for the player scores to 0 through the press of a key. To illustrate the instruction of the game I added a new scene, which appeared between the opening screen and the game. I used the mouse button rule and a change scene command to switch move between the three scenes. In the “Instruction” scene I displayed text which illustrated which buttons to use to control each player and what button to press to serve the ball. In the game scene I used a command upon every game restart a message would appear which reminded the players how to serve the ball. To finish things off, I added a rule which made the “Player 1” and “Player 2” collide and bounce off the wall so they wouldn’t disappear off the screen when the user is controlling them. In the physics section I altered their bounciness and set it to 0 so they wouldn’t bounce off the wall but only be restricted by it. In the ball’s physics I changed it’s friction to 0 so the ball would maintain it’s velocity after bouncing off an object.