

In this assignment the goal is to complete an angry birds clone game exhibiting the behaviors shown in the image above.

Please note: This is a graded assignment. This is a draft of the assignment made available before course work submission is open, only minor changes may occur.

Where do I submit?

There are eight graded assignments like this on the course. This one and three more make up the midterm assignment and are submitted in week 10. Nevertheless, we strongly recommend that you complete this graded assignment in the week it has been assigned and not wait for the midterm submission. You need to master the material included in these before you continue with the next weeks. Once you complete it, save it somewhere safe, do not share online using the webspace and upload it with the others when prompted in week 10.

Steps to complete

Start by downloading the template attached. The template code contains the sketch.js file which ties everything together and the physics.js file which contains physics-specific functions, moved to a separate file so that your code is cleaner. You'll write most of your code in the physics.js file.

angryBirdClone-TEMPLATE

☐ Creating a propeller system


☒ Step 1:

Amend the `setupPropeller()` function in `physics.js` to setup a static body of type rectangle at location (150, 480) of size (200, 15), similar to how the ground is created. Use the global variable `propeller` for this. The initial angle is equal to the global variable `angle` which we provided. Add the propeller to the world. *physics.js - line 18-22*

☒ Step 2:

In the `drawPropeller()` function set the angle of the propeller equal to the global variable `angle`. Set the angular velocity equal to the global variable `angleSpeed` already provided for you. Update the variable `angle` by `angleSpeed`. This will make sure the propeller actually moves at a rate of `angleSpeed` between frames. Draw the propeller using the `drawVertices()` helper function just like in the `drawGround()` function. *physics.js - line 26-33*
In `sketch.js`, amend the `keyPressed()` function so that when the left arrow is pressed, the angle speed is incremented by 0.01. Do the opposite when the right arrow is pressed. If you've done things right when you start your sketch you should be able to control the propeller on screen using the left and right arrows. *sketch.js - line 59 & 62*

○ Step 3:

↳ When 'b' is pressed a bird is created wherever the mouse is by calling the `setupBird()` function. *Sketch.js - line 70* 

Study that function. At the moment the program does not draw birds.

Amend the `drawBirds()` function to loop over the birds array and draw them using the `drawVertices()` helper function. *physics.js - line 47-49*

Use the `isOffScreen()` function to check if the bird has left the screen and, if it has, remove it from the physics world using the `removeFromWorld()` helper function provided and from the array. *Sketch.js - line 99-103*

Remember to also decrement your for-loop counter in order not to break your code! Pressing 'b' should now create a new bird where the mouse is which flies away upon impact with the moving propeller. *Sketch.js - line 102*

○ Creating a tower of boxes *stuck here*

○ **Step 4:** Amend the `setupTower()` function to create a tower of boxes as shown in the picture above. Tower should be six boxes high, three boxes wide. Each box should be of size 80x80 pixels. (Hint: Create a nested for loop and push bodies of type rectangle on the boxes array provided). Also push a random shades of green onto the colors array. We'll use those colors to draw the boxes later. Remember to add the boxes to the world.

○ Step 5: In the `drawTower()` function loop over the boxes array and draw each box using the `drawVertices()` helper function. Remember to use the random colors you created inside the colors array. You should now see a tower of boxes in different shades of green like in the image above.

○ Creating a slingshot

○ Step 6: Amend the `setupSlingshot()` function to initialise the global variable `slingshotBird` as a body of type circle in a similar place as shown in the image above. Give the circle zero friction and a restitution of 0.95. Set the mass of the `slingshotBird` as ten times its original mass, just like we have done for each of the birds in the `setupBird()` function.

○ Initialise also the global variable `slingshotConstraint` as a constraint that behaves and looks like the one shown above. Give it a stiffness of 0.01 and damping of 0.0001.

○ Remember to add both the bird and the constraint to the world and make sure they appear where they are in the image above.

○ Step 7: Amend the `drawSlingshot()` function and use the `drawVertices()` and `drawConstraint()` helper functions to draw the slingshot bird and slingshot constraint. If you've done things right you should now be able to control the slingshot with the mouse. Drag to extend, release mouse

to release bird. Pressing 'r' resets the slingshot.

- ☐ Step 8: Implement one of the ideas for further development.

- ☐ Ideas for further development
- ☐ Turn it into a game. Create a countdown. The player has a 60 seconds to remove all boxes from the screen. If they succeed, they win. If they fail, looping stops and a GAME OVER message is displayed.

- ☐ Make it your own by changing the colors and style.

- ☐ Add another physics object that adds to the gameplay in a fun way.

Marking rubric

Step 1 - [1 point]: Propeller in right place approx.

Step 2 - [1 point]: Propeller moves with left/right arrow.

Step 3 - [2 points]: Birds are drawn when pressing 'b' and interact with propeller. Removed from world and array when off-screen.

Step 4 - [0 point]: (points transferred to next step)

Step 5 - [3 points]: Box tower is in the right place approx. Has dimensions of example provided. Boxes use multiple colors.

Step 6 - [0 point]: (points transferred to next step)

Step 7 - [3 points]: Slingshot appears in right place. Has the right stiffness. Slingshot bird has enough mass to demolish tower.

Step 8 - [3 points]: One of the ideas for further development has been implemented in an impressive way. Full points for learners that challenged themselves.

Step 9 - [4 points]: Code presentation: formatting, comments, variable naming.