Assignment 4: Atari Breakout!

WebGL without a framework is hard. But this makes it even more rewarding when we create cool stuff!

In class, we learned how to draw multiple objects (rectangles) with different properties (colors and offset). We also made the rectangles move and we added the infamous fish!! In this assignment, we will create a simple but fun video game based on the things we learned. In the game, we will recreate the famous Atari Breakout! game in a Gameboy Advance (see below).

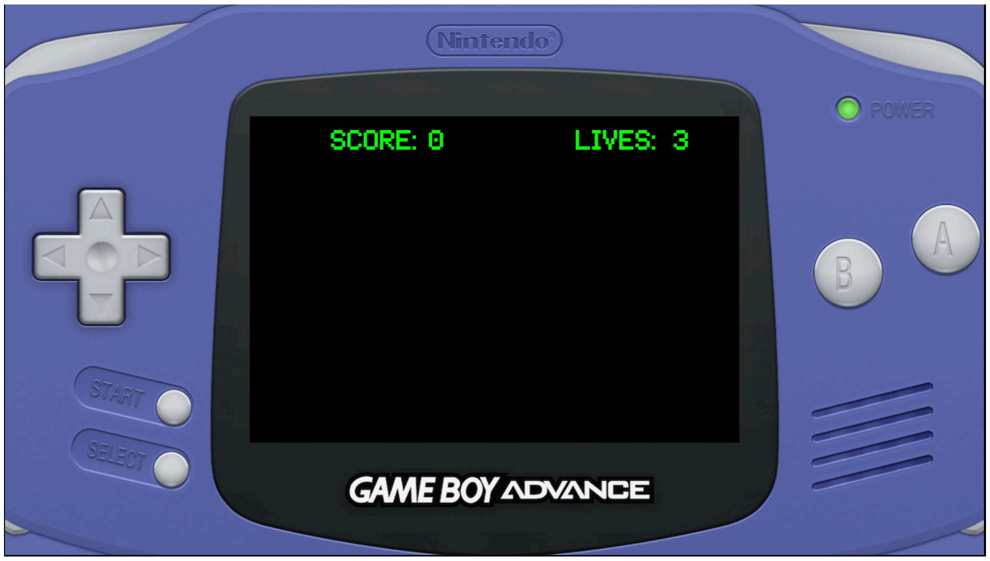
By the way, you can play a Google Easter egg game based on Atari Breakout here: `https://elgoog.im/breakout/`

Starter code: Please sync your fork and then use `04/index.html` as the starter code!

Part 1: Change createRectangle to createPaddle (10 Points)

In class we created the `createRectangle` method to add multiple rectangles to the `OBJECTS` list. Now, we need two paddles to play the game so let's rename the `createRectangle` method to `createPaddle`. Also, we are working on the small display of Keen's computer wrist and need to squeeze the paddles a bit. The following vertices look good so please replace the old code as follows:

```
vertices = new Float32Array( [
                              -0.1,  0.05, 0.0, // V0      // 0
                              -0.1, -0.05, 0.0, // V1, V4  // 1
                               0.1,  0.05, 0.0, // V2, V3  // 2
                               0.1, -0.05, 0.0  // V5      // 3
                            ] );
```



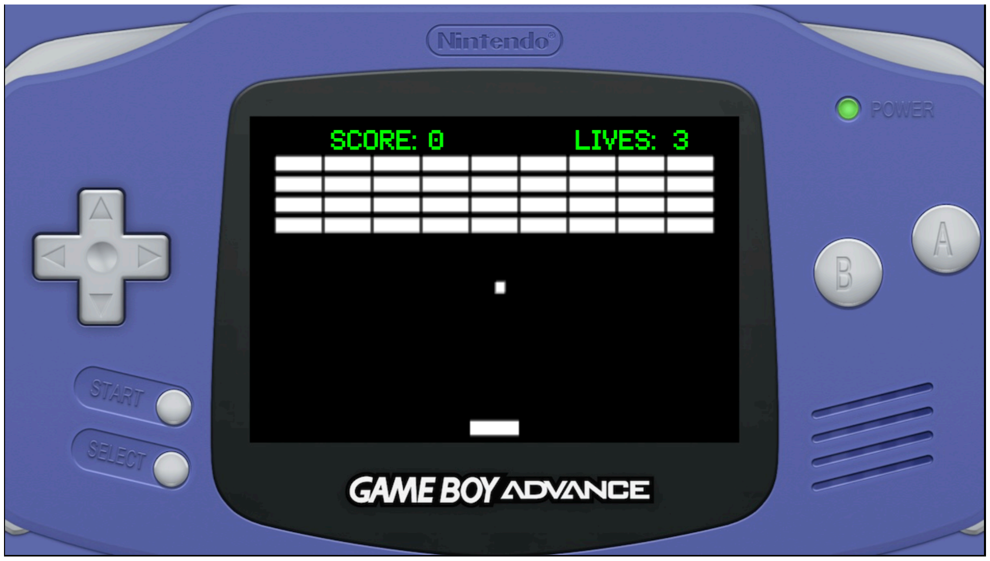
But the paddles do not appear yet, because we will need to create the ball first!

Part 2: Implement createBall (30 Points)

In the starter code, `createBall(color,offset)` is an empty method. Please implement it to render a single point at `0,0,0` using an indexed geometry. Now, please make sure that we render that single point by specifying a `gl_PointSize` of `8` pixels in the vertex shader (make sure it is a float).

So far, so good. However, the ball will not appear since our rectangles are rendered as `gl.TRIANGLES` and here we will need `gl.POINTS`. The best way to address this is to change our geometry representation to include the rendering primitive. We now need to include another element: the rendering primitive for `createBall`, `createPaddle` and `createBrick`. (Add the primitive variable at index 5 of the return array, for example: Change `[v_buffer,i_buffer,color,offset]` to `[v_buffer,i_buffer,color,offset,gl.POINTS]`.

After, we will need to change the rendering loop `animate` to load the rendering primitive for each object and modify the `drawElements` call accordingly and our paddles and ball will appear!

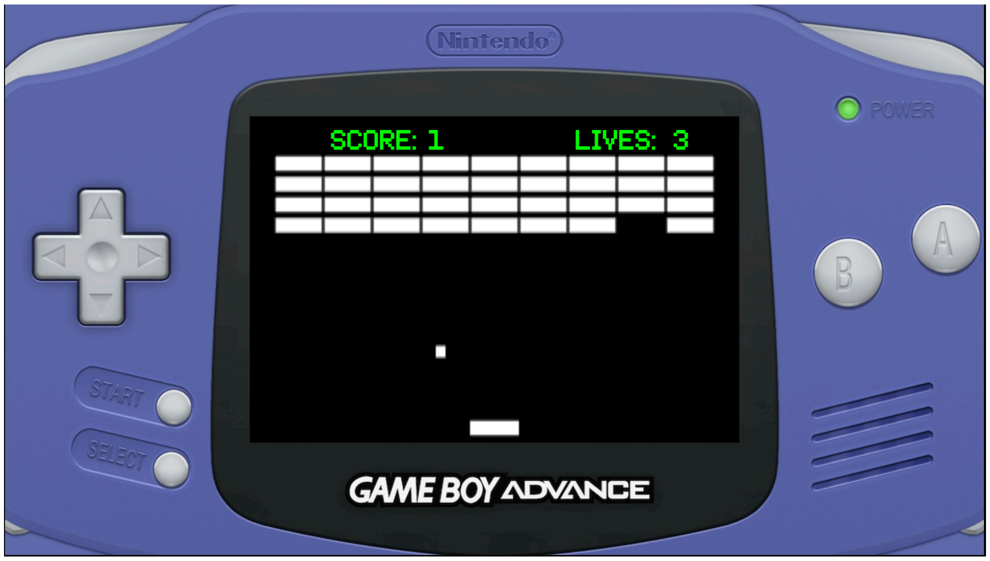


Part 3: Add interaction (20 Points)!

But nothing moves yet! For this part, please modify the last method in the code: the `window.onkeydown` callback. Please implement movement of the Keen's paddle—the one in the bottom—when you press `ArrowLeft` or `ArrowRight`.

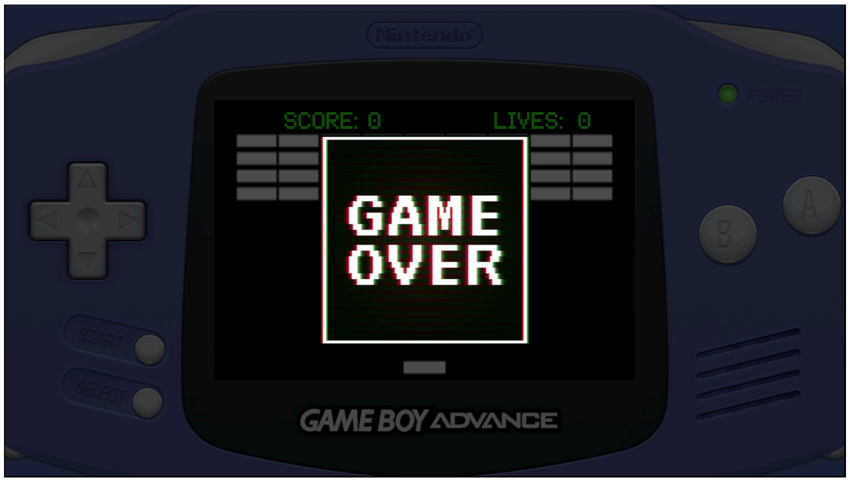
Make sure that the paddle does not leave the viewport and change the offset by a small amount for each press. After this, you should be able to control the paddle.

Part 4: Study updateBall() and call it! (10 Points)
The starter code includes the `updateBall` method. Please study the code and be ready to explain it when submitting the form. Also, the method needs to be called every frame. Please modify `animate` to call it at the proper time.



With these changes, you can control your paddle while the ball moves.

Part 5: Study createBricks() and make it colorful! (10 Points)
The starter code also includes the `createBricks` method. Please study the code and be ready to explain it during form submission. Make each row of bricks a different color or use random colors!!



Part 6: Study resetBall(), loseLife() and score() (10 Points)
There are two more methods to study and to explain: `resetBall`, `loseLife` and `score`.

Part 7: Submit the Form! (10 Points)
Finally, please push everything to your fork and double-check Github Pages! As usual, please send a pull request and then fill in the form at `https://cs460.org/assignments/04/` but don't submit it yet if you plan on doing the bonus (aka. the fun part!).

Bonus (Total 33 points):

Bonus Part 1: Add Sound Effects! (5 points)
Add some cool sounds for instance when the ball is reflected and some background music or some effects when you win or loose. You can use the `<audio>` tags for that.

Bonus Part 2: Speed up! (8 points)
The game is relatively boring right now - make the ball move faster with time :).

Bonus Part 3: Shrinking/Growing Paddles! (9 points)
Changing the paddle size could also add some entertainment - be creative :).

Bonus Part 4: Brick Mania! (11 points)
Modify the createBricks function to add more rows of bricks, you can also make the size of the bricks to be smaller or larger and add more bricks to a row.

SUPERBONUS: Up to 33 extra points for the most creative solutions!!
Feel free to customize any gameplay or visuals!