Here is the discussion forum for this part of the course. You can post your comments and share your creations here, and of course ask questions.

Let us suggest some topics of discussion and optional projects:

SUGGESTED TOPICS

- Did you already know about event handling in JavaScript or was this new to you?
- How can we more efficiently handle multiple key presses together with mouse button clicks, mouse moves, etc? The course gives all the basics but there may be other more elegant ways.
- Did you know about a gamepad API? Read this if you want to try it.
- How do we make a responsive game that works in a canvas?

OPTIONAL PROJECTS

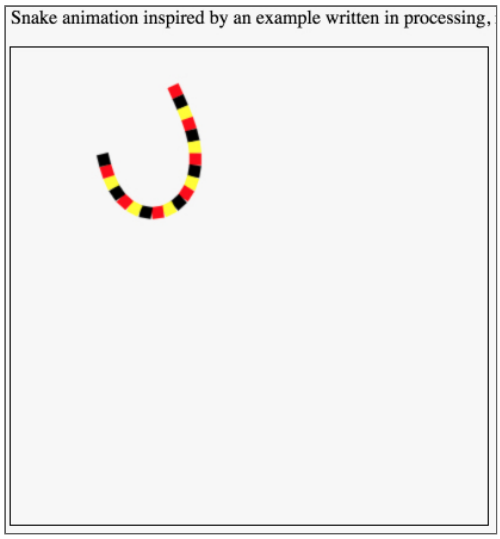
Here are a few project ideas. Your classmates and the team who prepared the course will be glad to try them and offer feedback. Please post URLs in this discussion forum. These projects are optional, meaning that they won't be graded.

- **Project 1 (easy)**: Make one of your drawings move in all directions (left, right up, down, then diagonals) using the arrow keys.
- **Project 2 (a bit harder)**: Make an animated chart. When the page is loaded, the chart "grows" until the chart bars reach their "normal" value. Another variant is to use animated colors or shadows in your chart.
- **WE ALSO PREPARED A "Snake" challenge, look at the bottom of this page!**
- **Project 3 (a bit harder)**: Make your monster follow the mouse + open its mouth and change color when we click on a mouse button. If you manage to make it scream, it's even better (use a hidden audio element and call play(). Advanced users may want to take a look at the howler.js JavaScript library that loads sound samples in memory and plays them on demand).
- **Project 4 (advanced)**: On the Web, look for JavaScript functions for detecting collisions (circle/circle or rectangle/rectangle), and try to make a small game in which

your monster must "eat" some balls that bounce on the screen. Every 5s new balls appear on the canvas. Make your monster go towards the mouse pointer. You can use var angle = Math.atan2(dy, dx); in order to compute the monster angle, dx and dy = difference between the monster and mouse positions.

- **Project 5 (easy to intermediate)**: Put into practice what you've learned about the responsive canvas and develop an example of your own. The difficulty may vary depending on the things that are drawn or animated.

THE "SNAKE" CHALLENGE



Just for fun, and for those of you who like a challenge, I adapted a small snake animation from a version written in processing to HTML5 / canvas / requestAnimationFrame.

Try it on JsBin!

Please make some improvements to the snake by adding a nice head and tail, changing the colors, adding a background, etc. Be creative and tweak this according to your artistic taste :-)

POST YOUR CREATION AND COMMENTS IN THE FORUM: there is a dedicated thread for that!

It's also a very interesting example that shows the power of 2D transformations + interesting use of `ctx.save() / ctx.restore()`, as each segment of the snake is always drawn at a fixed position: only the coordinate system of the previous segment is translated/rotated -> no complicated computations!

The challenge is on! I hope you'll enjoy it!

What you could try:

- Be creative!

- Add some interesting features (forked tongue, head, tail, snake tortoise shells...), longer, faster, slower, drunk, etc
- Make the snake chase the mouse (the snakes moves slower, so *it tries* to follow the mouse)
- Animate not one snake, but many
- Make a game of it
- You can just change the loop of drawSnake to reverse the stacking order of segments. Try with this:
  `for(var i=x.length-1; i >=0; i--)` in drawSnake()

  As shown in this example, if the snake's body crosses over itself, it will pass "on top", not "under", **producing a nice "elastic effect". Do you understand why this effect is produced?**