Santa Rush

Final project pitch

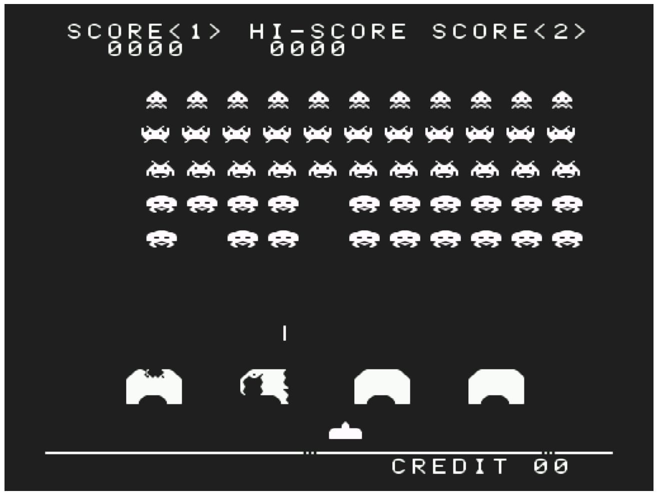
Marie-Eve Cousineau / Final project pitch / CART 253 / Autumn 2017

This project's visuals will mostly be developped for another class' final project (CART214 - Visual Communication).

Summary

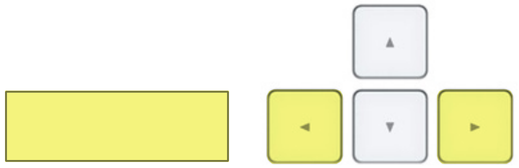
“Santa Rush” is a game based on Space Invaders. The controls and the way the elements behave and are displayed will be inspired from it. As you may have guessed, the game’s theme is Christmas (Because we have to get into the spirit!) The space invaders, at the top, are replaced with Santa’s little helpers, and of course, the goal is not to destroy them, because they’re not your enemies! So there will always be around 12, but this number might increase or decrease once I start creating the game. Instead of throwing lasers at you, they’re dropping toys, and please don’t let them crash on the floor because they are going to break and children won’t have gifts in the morning. If it wasn’t clear enough already, you indeed have to catch them with Santa, whom you control at the bottom of the window as to replace the paddle. If you do not catch a toy, you lose a “life”, and you have three lives. If you lose them all, it’s game over. These lives, instead of being represented as hearts like in any video game, will be represented by the icon of a toy as well, meaning that you can only break three. To win the game, you have to “stay alive” until the timer ends. The game lasts about 5 minutes and the further you get, the harder it gets, because the Christmas elves are producing more because the Christmas rush becomes more and more intense (where the name comes from, obviously). The background will be an eerie Christmas-like scenery, with snow and Christmas decorations.

Inspiration / interactivity



SPACE INVADERS

controls



The elements interact between each other a bit like Space Invaders, as said before. The only element you have to control is Santa with the arrow keys, whom replaces the paddle at the bottom. At the top, there is going to be two moving rows of six Christmas elves dropping toys. Where and when the toys are dropped will be totally random. The two rows will move seperately from one another. **But because the goal is not to destroy them, the rows will not go lower and lower, they will always stay on the same Y axis, and new ones won't be generated.**

Refer to the summary for more details of the game

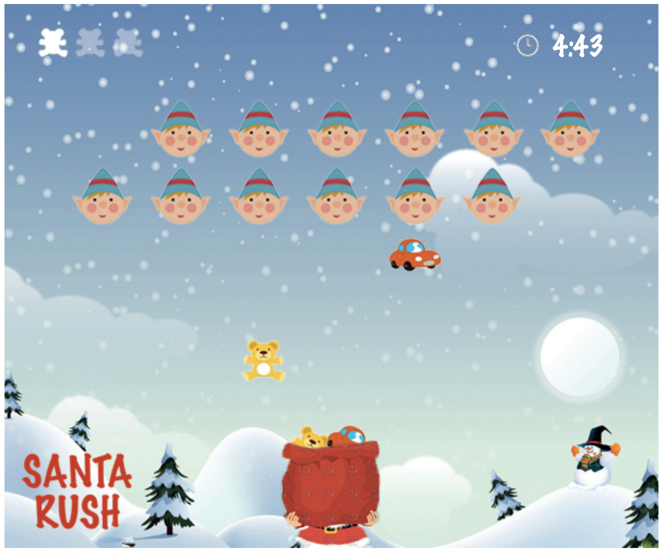
Inspiration / visuals

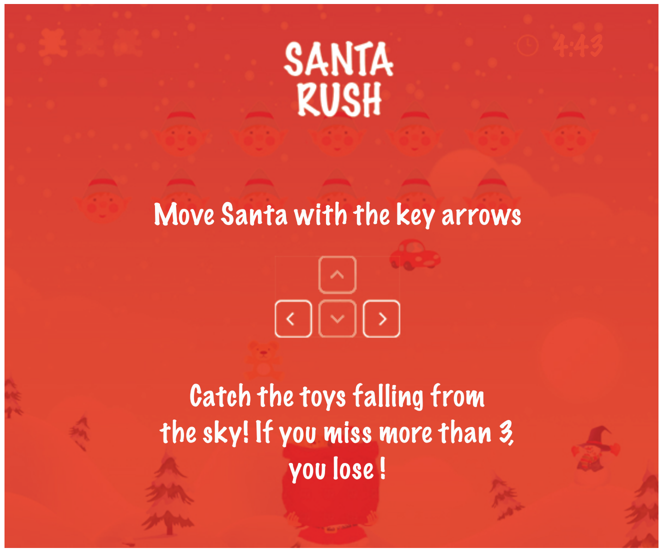
The visuals will be based on the Christmas theme, and inspired from phone app game designs or even from Facebook stickers (so a very light and simple but "adorable" kind of style, something that illustrates Christmas spirit).

COOKIE RUN
sweet escape adventure
Hi
Bye
SANTA'S SHOP
+100 $0.99
+250 $1.99
+500 $3.49
+10 SURPRISE GIFT $0.99
+10 REMOVE LINES $0.99
MENU
PAUSE
SHOP
INVITE YOUR FRIENDS!
JINGLE GRUMPS

Media

The visuals will be based on the Christmas theme, and inspired from phone app game designs or even from Facebook stickers (so a very light and simple but "adorable" kind of style, something that inspires the joy of Christmas spirit). The following mockups represent roughly what the game will look like.





Move Santa with the key arrows

Catch the toys falling from the sky! If you miss more than 3, you lose !

Main screen : This is the screen where the game is running. On the top right corner, you can see the countdown. On the other side is your number of remaining chances. The elves are going to drop 3 different toys.

Text screen : The screen is going to look like that either when it shows the controls at the beginning, when it says game over or when you win. A bright red screen like this means the game is not running.

Technical approach

`keyPressed()`

The keyPressed function will be used to check if the right or left arrow have been pressed, which will make Santa move.

`keyReleased()`

The keyReleased function will tell Santa to stop moving when the arrow keys are released.

`loadImage()`

Of course, we will need to load our images with the loadImage fonction (each element will be an image).

`constrain()`

This function defines the limits of Santa's movement. We will tell him that he cannot go out of the screen. We will also constrain the elements' movement to X axis only (except the toys, which will be moving on the Y axis only).

`collide()`

The collide function, just like Pong once again, will help us define when Santa and the dropped toy collide, because when it doesn't, the player loses a life.

`millis()`

The millis function will allow us to create the timer which will end the game after 5 minutes if the player still has at least one life.

`array()`

The array function will be used to store each elf's position in X and Y, and we will use it to tell the program from where the toys should be dropped (just like the Griddies exercise).

`Class Santa {}`

This class will be defined for Santa whom acts like a paddle. There is only one.

`class Elf {}`

This class will be defined for Santa's little helpers, at the top of the screen. There will be around twelve, distributed in two rows.

`Class Toy {}`

This class will be defined for the toys dropped from the top of the screen. There will be more and more generated as the game approaches the end.

Technical research

How to make text appear and disappear?

To do so, you would have to put this code in draw :

```
line = line.equals(firstText)? secondTex
t:firstText;
```

And then, put the following in setup :

```
line = firstText;
font = createFont("arial", 20);
textFont(font);
fill(255);
```

You integers have to be called at the beginning :

```
String firstText = "Press SPACE to begin";
String secondText = "Press the left and
right arrow key to move Santa left and
right";
String line = "";
```

How to make an image appear and disappear?

To do so, you simply have to create a boolean (boolean showImage, example), and to make sure that somewhere in the draw, along with your false condition which puts the image out of screen, you also display the image:

```
  if (showimage) {
image(a, X, Y);
}
```

How to add Christmas music?

To do so, you have to put this code in setup :

```
file = new SoundFile(this, "sample.mp3");
file.play();
```

You integers have to be called at the beginning :

```
import processing.sound.*;
SoundFile file;
```

So it's just like when you load an image.

How to create falling snowflakes?

Found here : http://solemone.de/demos/snow-effect-processing/

```
int quantity = 300;
float [] xPosition = new float[quantity];
float [] yPosition = new float[quantity];
int [] flakeSize = new int[quantity];
int [] direction = new int[quantity];
int minFlakeSize = 1;
int maxFlakeSize = 5;

void setup() {

  size(800, 350);
  frameRate(30);
  noStroke();
  smooth();

  for(int i = 0; i < quantity; i++) {
    flakeSize[i] = round(random(minFlake-
Size, maxFlakeSize));
    xPosition[i] = random(0, width);
    yPosition[i] = random(0, height);
    direction[i] = round(random(0, 1));
  }
}
```

```
void draw() {

  background(0);
  for(int i = 0; i < xPosition.length; i++)
{

    ellipse(xPosition[i], yPosition[i],
flakeSize[i], flakeSize[i]);

    if(direction[i] == 0) {
      xPosition[i] += map(flakeSize[i], min-
FlakeSize, maxFlakeSize, .1, .5);
    } else {
      xPosition[i] -= map(flakeSize[i], min-
FlakeSize, maxFlakeSize, .1, .5);
    }

    yPosition[i] += flakeSize[i] + direc-
tion[i];

    if(xPosition[i] > width + flakeSize[i]
|| xPosition[i] < -flakeSize[i] || yPosi-
tion[i] > height + flakeSize[i]) {
      xPosition[i] = random(0, width);
      yPosition[i] = -flakeSize[i];
    }
  }
}
```