

15-104 Introduction to Computing for Creative Practice

Fall 2022

29 Platform Game

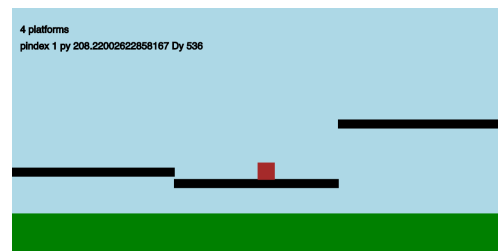
Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

1



A platform game

- The platforms scroll right to left.
- The character will always remain in the middle of the canvas.
- The character should jump when a key is pressed.
- If the character is in the air, it will fall.
- If the character lands on a platform, it is safe.
- If the character falls to the bottom, it will “die”.
(but in our game, it will re-emerge, falling from the sky)

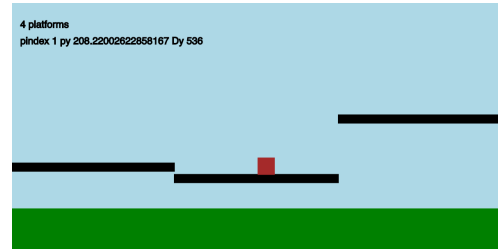


2

Platform

- Each platform will be an object with three data fields:
 x: horizontal location (top left)
 y: vertical location (top left)
 w: width of platform
- Methods:
 right – returns the x location of the right end of the platform

(All platforms will have a thickness of 10 pixels.)



3

Platform Code

```
function newPlatform(px, py, pw) {
  var p = {x: px, y: py, w: pw,
           right: platRight};
  return p;
}

// compute the location of the right end of a platform
function platRight() {
  return this.x + this.w;
}
```

4



Setting up the game

```
var platforms = [];      // array of platforms

function setup() {
  createCanvas(600, 300);
  // first platform, just off the canvas:
  var pl = newPlatform(600, 200, 200);
  platforms.push(pl);
}
```

5



Drawing the game

```
function draw() {
  background("lightblue");           // the sky
  fill("green"); stroke("green");
  rect(0, height - 50, width, 50); // the ground
  fill(0); stroke(0);
  for (var i = 0; i < platforms.length; i++) {
    var p = platforms[i];
    p.x = p.x - 1;
    rect(p.x, p.y, p.w, 10);
  }
  // UPDATE PLATFORM ARRAY HERE (next slide)
}
```

6

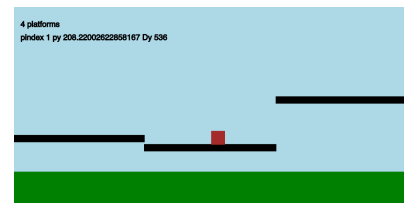
Update Platform Array

```
// if first platform is offscreen to left, remove it
if (platforms.length > 0 && platforms[0].right() < 0) {
  platforms.shift();
}
// if last platform is totally within canvas, make a new one
var lastPlat = platforms[platforms.length-1];
if (lastPlat.right() == width - 1) {
  var p = newPlatform(
    width,           // left edge just off canvas
    floor(random(50, 225)), // height above bottom
    200); // all platforms have width 200 for now
  platforms.push(p); // add to our array of platforms
}
```

7

Adding the Character

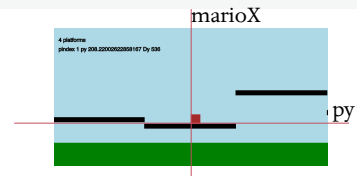
- We always draw the character in the middle of the canvas, so we only worry about the vertical (Y) coordinate.
- We need to find which platform is currently in the middle of the screen, so we do a *linear* search to find it.
- Then we move the character based on whether it is above, on, or below the relevant platform.
- The character automatically “wraps around” when it falls, which means we need no interaction to do some testing.



8

Finding platform in middle

```
var i = 0;
var marioX = width / 2;
while (platforms[i].right() < marioX) {
  i += 1;
}
var py = platforms[i].y; // height of middle platform
```



(There's a bug here!)

<code>platforms</code>	is the array of platforms
<code>platforms[i]</code>	is the i^{th} platform in the array of platforms
<code>platforms[i].right()</code>	is the method of the i^{th} platform in the array of platforms that returns its right coordinate.
<code>platforms[i].y</code>	is the y-value of the i^{th} platform in the array of platforms

9

Moving and drawing the character

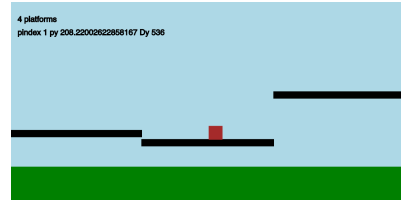
```
var marioY = 0; // new global variable, start "mario" at top
-----
if (marioY == py) {
  marioY = py; // on the platform, don't change marioY
} else if (marioY == height) {
  marioY = 0; // "dies", re-emerge at top of canvas
} else {
  marioY += 1; // otherwise, "fall" one pixel
}

// draw the "mario"
fill("brown");
stroke("brown");
rect(marioX, marioY - 20, 20, 20);
```

10

Adding a Jump feature

- To implement jumping, we start by adding `marioDy`, which is the velocity of the “mario” character.
- New global variable:
`var marioDy = 0;`
 for the vertical velocity of the character:
 zero: floating
 positive: falling
 negative: rising



11

Rising and Falling

```

if (marioY <= py) { // above or on platform
  marioY = min(py, marioY + marioDy);
} else { // below platform
  // we don't want upward movement
  if (marioDy < 0) {
    marioDy = 0;
  }
  marioY = min(height, marioY + marioDy);
}
if (marioY >= height) {
  marioY = 0; marioDy = 0;
}

```

If `marioDy > 0`, then “mario” is falling (y will increase).
 If `marioDy < 0`, then “mario” is rising (jumping).

12

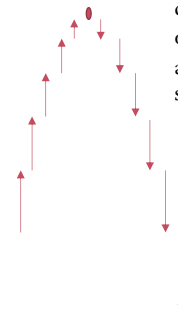
Updating the velocity

```
// accelerate "mario" with gravity
marioDy = marioDy + 1;
```

Making “mario” jump:

```
function keyPressed() {
  marioDy = -10;
}
```

Example of how `marioY` changes due to `marioDy` over repetitions of `draw` assuming a key press sets `marioDy` to `-5`.



13

Fix This

- When the game starts, the character falls to the height of the first platform even though it's not there yet. Fix this.
- When the character lands on the platform, `marioDy` continues to increase. Fix this.
- When the character jumps, if its right edge touches the next platform, it still falls. Fix this.
- All platforms are 200 pixels wide. Fix this so that each new platform is randomly 50, 100, 150 or 200 pixels wide.

14