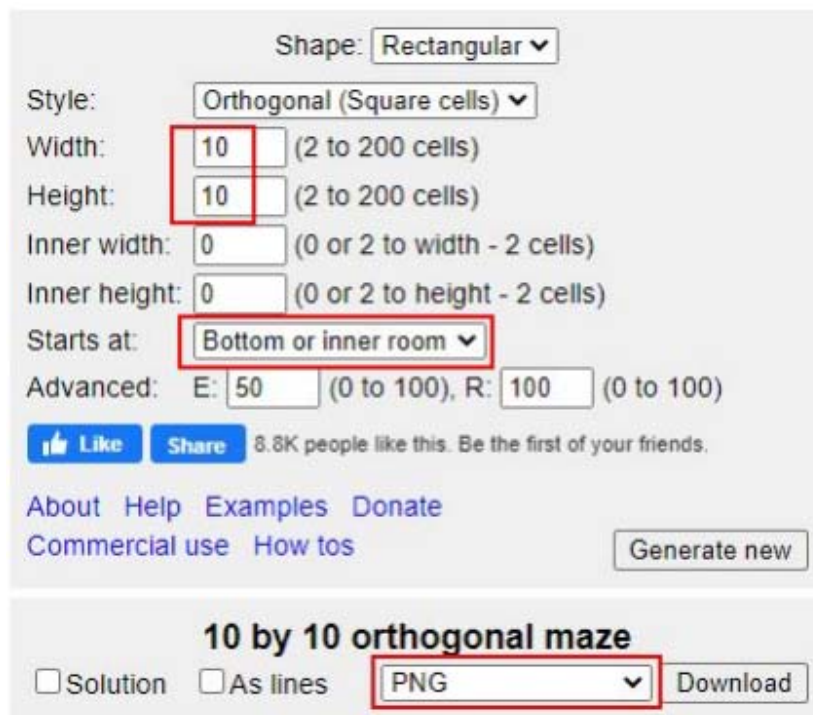


## Overview

You are required to create a Maze Game using MIT App Inventor 2. You are expected to demonstrate good use of MIT App Inventor features and creativity.

## General guidelines of the Game

1. Generate a random maze from the website:  
<http://www.mazegenerator.net/>
2. Use the following set of parameters:



The screenshot shows the configuration interface for the Mazegenerator.net website. The parameters are as follows:

- Shape: Rectangular (dropdown)
- Style: Orthogonal (Square cells) (dropdown)
- Width: 10 (input field, range 2 to 200 cells)
- Height: 10 (input field, range 2 to 200 cells)
- Inner width: 0 (input field, range 0 or 2 to width - 2 cells)
- Inner height: 0 (input field, range 0 or 2 to height - 2 cells)
- Starts at: Bottom or inner room (dropdown)
- Advanced: E: 50 (input field, range 0 to 100), R: 100 (input field, range 0 to 100)

Below the parameters, there are social media buttons for 'Like' and 'Share', and a 'Generate new' button. At the bottom, there is a section for '10 by 10 orthogonal maze' with options for 'Solution', 'As lines', 'PNG' (selected in a dropdown), and a 'Download' button.

Download the file and use it as the background image of the canvas.

3. Your maze should have an Entrance and an Exit to represent the start and end of the game.
4. You can use the emoji from <https://emojipedia.org/> as the icons in your game (e.g. exit door, player, obstacles and etc.)
5. The user should be able to control the movement of Player1 by touching the canvas such that Player1 will move either vertically or horizontally towards the point of touch.

## 6. Possible enhancements of the game

- Allow users to restart the game.
- Existence of bonus or obstacles
- Multi-level of difficulties
- Scoring / timing

## **Requirements**

### Compulsory features

You are required to put your **Name, Class and Class no as the title of the main screen.**

Your game must have ALL of the following features:

1. Drawing Canvas to display graphics.
2. Image Sprites and their interactions with user.
3. Labels to display textual output.
4. User of conditional blocks (e.g. if-then-else) and Looping blocks (e.g. while test-do / for loop)
5. Use of global or local variables.
6. Use of procedures or functions.
7. Use of clock.

### Optional features

1. Multiple screens
2. Database storage
3. Sound effects

Bonus marks would be given if the use of these features (or any other features not listed) is found to be useful and appropriate.

### Other requirements

You should make use of the features of App Inventor appropriately. Marks may deducted if you apply feature(s) in an unsuitable situation.

## Marking

The project mark constitutes **15%** of the final subject mark. Marking is based on the following:

- Content: **6%** (amount and relevance to the theme, presence of all compulsory features)
- Techniques: **5%** (correctness, appropriate use of features)
- Creativity **4%** (fun to use, innovative)

## Submission

### File to be submitted

You need to submit the .aia file.

Save your project as “**TermPrj1\_class\_class no.aia**” using your actual *class* and *class no.* for the filename.

**.apk file is NOT accepted and should not be submitted.**

### Deadline

23:55 on 31 Dec 2022

### Procedure

You are required to submit the file via e-class. Do NOT submit disks, CDRs or via Teams / e-mail.

### Notes

- Log-in the eClass system well before you submit the file so that any problems can be reported to your teachers as soon as possible.
- Individual problems related to the login process are NOT accepted as a reason for late submission.
- You have to take the risk of busy network traffic if you submit your file near the deadline. Any kind of delay is NOT accepted as a reason for late submission.
- You MUST submit the .aia file but not any other types of files.
- You can re-submit your file. However, only the latest file you submitted will be considered. The date and time of submission also follow the latest file you submitted.
- You MUST make a copy of the .aia file yourself. You may be required to provide the file again if necessary.

## Appendix

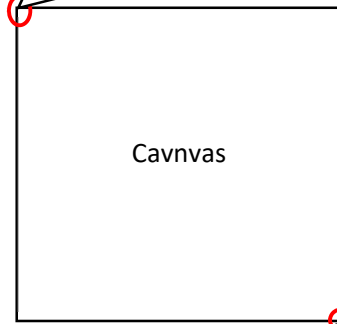
You can refer to the following suggested directions for your game.

1. How to capture the user touching position and move the player icon accordingly?

Return the x-coordinate and y-coordinate of the touching point in the Canvas

```
when Canvas1 .Touched
  x y touchedAnySprite
do
```

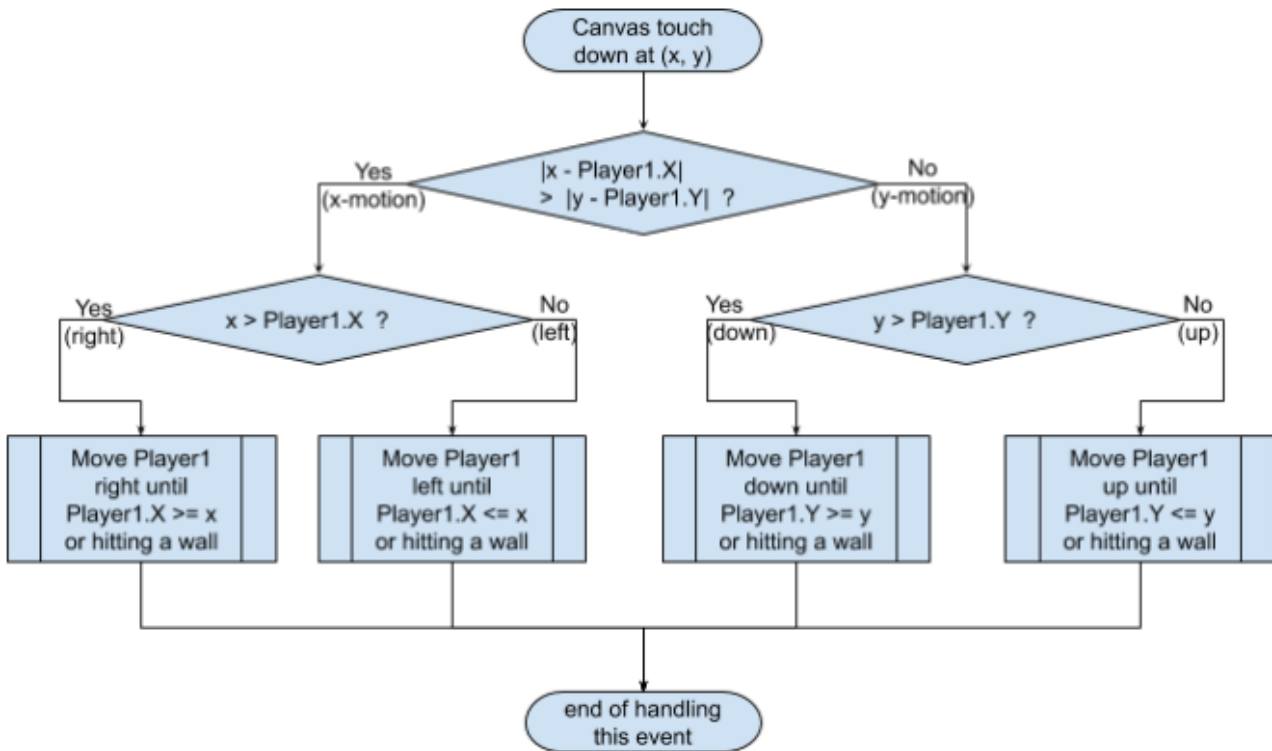
This is the position of (0, 0)



This is the position of (100, 150) if the canvas is 100 pixels width and 150 pixels long.

2. How to detect if a move is a horizontal / vertical move?

You can check the absolute difference between the touching position and the position of the player. If the horizontal difference (difference between their x-coordinates) is larger than the vertical difference (difference between their y-coordinates), a horizontal move should be taken. Similar idea apply on vertical move.



$|x - \text{Player1.X}|$  means the absolute value of  $(x - \text{Player1.X})$ .

The definition of absolute value of function is

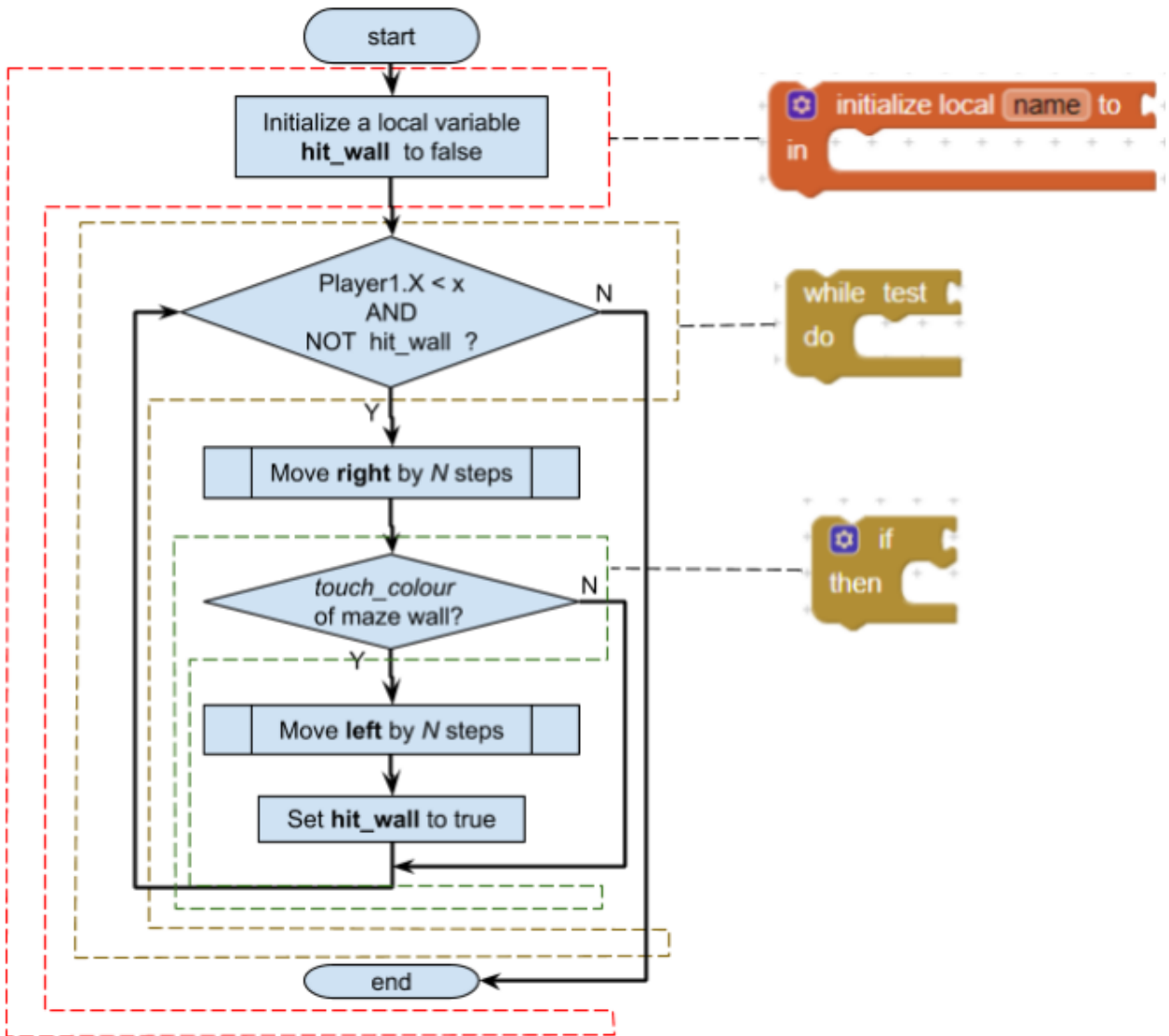
$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$$

You may use the following blocks to calculate the difference.



3. How to move Player1 to the right until it reach a wall or the touching position?

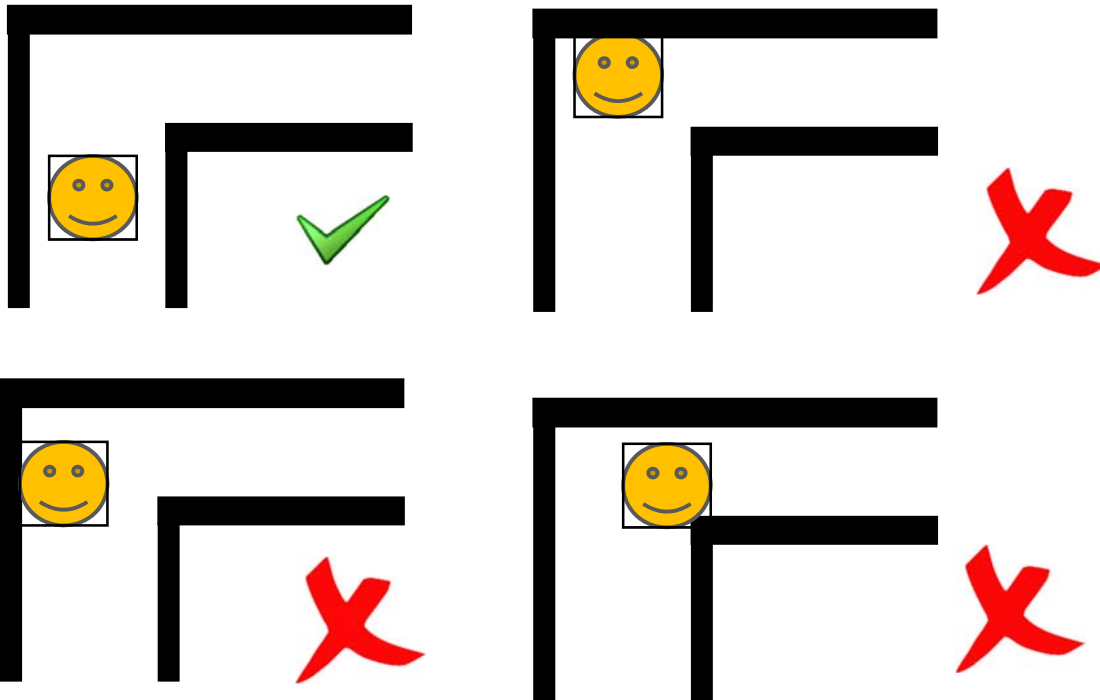
Move Player1 right and check Player1.X >= x or hitting a wall
--



Larger  $N$  makes it move faster, but should not be too large ( $<$  thickness of the wall). Otherwise, it may fail to detect a collision to the wall.

4. How to detect a collision to a wall?

We can check the color of the 4 corners of the icon Player1. If the color is equal to black (color of wall), collision happens.



```
call Canvas1 .GetBackgroundPixelColor
x Player1 . X
y Player1 . Y
```