Task Description

In this assignment, you will create a game in the Java programming language using the Processing library for graphics and gradle as a dependency manager. In the game, balls spawn and move around the screen and the player can draw lines to direct them into holes of the same colour. When balls are reflected off a player-drawn line it disappears. If a ball enters a hole of a wrong colour, score is lost and it respawns. Once all balls are captured by holes, the player wins.

Gameplay

The game contains a number of entities that will need to be implemented within your application.

Level

Each level is read from a t d be 576x640,
meaning each character i

are defined in files
layout” attribute of
ration file described
must have an
file.

does not need to
8x18=324 characters.
han this if they are not
as spaces at the end of
lines at the bottom).
, your program should
ust reflect balls off the
en.

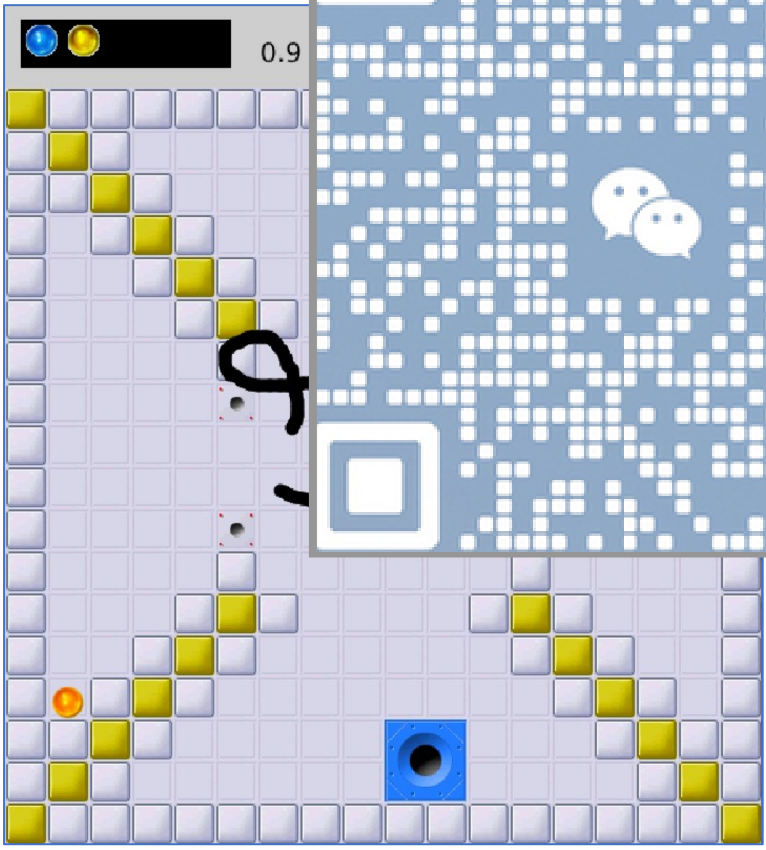
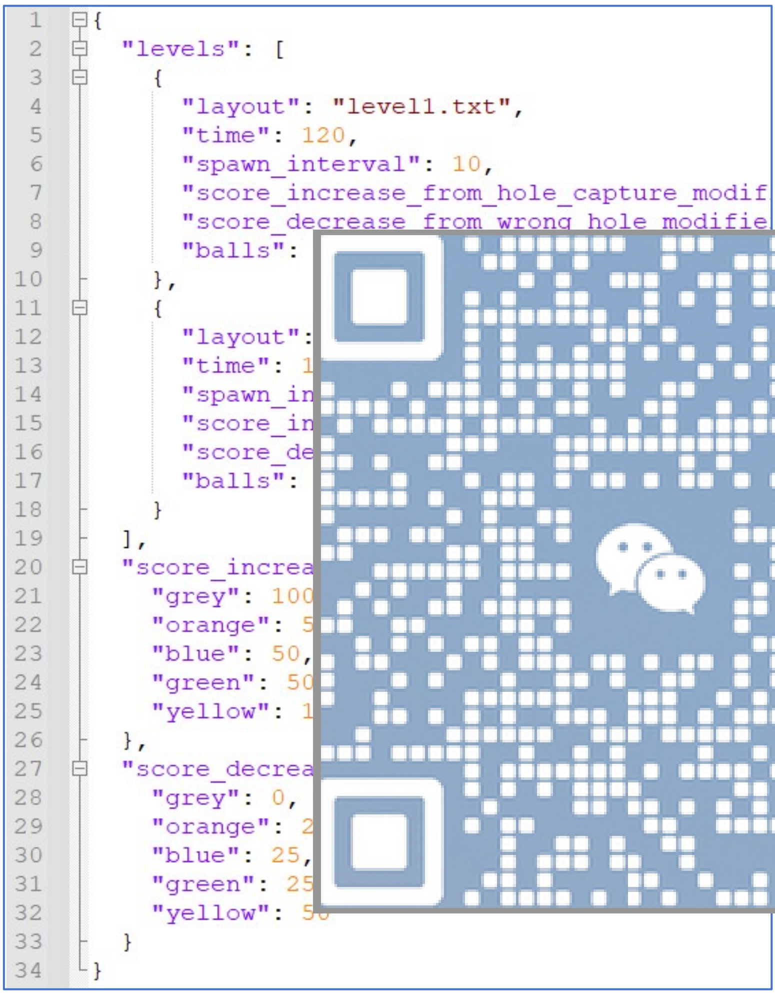


Figure 1.
The second provided sample level

There are 5 main types of characters that could be present in the file:

- X – denotes a wall (wall0.png). Balls reflect off this, but do not change colour. The game board does not need to be surrounded by walls – balls should reflect off the edge of the screen.

- 1,2,3,4: walls 1,2,3 and 4 respectively, as provided in the scaffold resources. When a ball hits one of these walls, it is reflected and changes colour to that of the wall.
- S – Spawner. Balls spawn from this location (one spawner is chosen randomly of all available spawners in the current level, each time a ball is ready to be spawned).
- H – Holes. The hole takes up 4 tiles, where the 'H' character is the one in the top left. The number in the character to the right of the H is the colour of the hole.
- B – Balls. Instead of spawning after the spawn interval, a ball may be present immediately from the level beginning, at a specific place on the board. The colour of the ball is denoted by the character to the right of the 'B'.
- Spaces – empty space, just ignore it (blank tile).



```
{
  "levels": [
    {
      "layout": "level1.txt",
      "time": 120,
      "spawn_interval": 10,
      "score_increase_from_hole_capture_modif
      "score_decrease_from_wrong_hole_modifie
      "balls":
    },
    {
      "layout":
      "time": 1
      "spawn_in
      "score_in
      "score_de
      "balls":
    }
  ],
  "score_increa
    "grey": 100
    "orange": 5
    "blue": 50,
    "green": 50
    "yellow": 1
  },
  "score_decrea
    "grey": 0,
    "orange": 2
    "blue": 25,
    "green": 25
    "yellow": 50
  }
}
```

Config

The config file is in located in config.json in the root directory of the project (the same directory as build.gradle and the src folder). Use a ad it. Sample config e provided in the

files will also be ot directory of the r, sprites or images

nd `wall x.png` will be sources folder rces/inkball/) or /main/inkball/).

file.

For each level, the following properties are provided in the config:

- `layout`: the level file containing the characters determining the position of tiles in the grid for walls, holes, spawners and initial balls.
- `time`: the maximum number of seconds this level should last for. If the time is exceeded, the player loses the level and it restarts. If the time is invalid (eg, non-integer or negative value like -1) or missing, there is no timer for this level.
- `spawn_interval`: the number of seconds in between when balls spawn.

- `score_increase_from_hole_capture:` The amount of units score is increased for each ball type when they successfully enter a hole.
- `score_increase_from_hole_capture_modifier:` Multiply the score values gained on this level by this modifier.
- `score_decrease_from_wrong_hole:` The amount of units score is decreased for each ball type when they enter the wrong hole.
- `score_decrease_from_wrong_hole_modifier:` Multiply the score values lost (when a ball enters a wrong hole) on this level by this modifier.

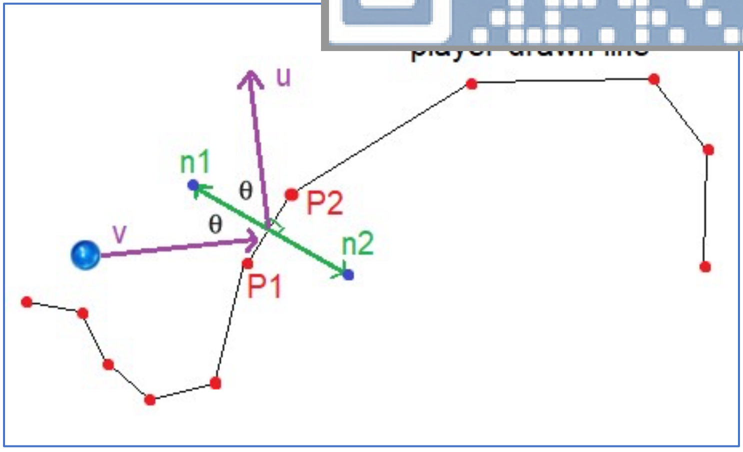
Balls ●●●●●

Balls may appear in the level layout file, as "B0", "B1", "B2", etc in which case they are spawned immediately in that location when the level begins. Alternatively, they may also be specified in the configuration file, which will cause them to be spawned at a spawner ● throughout the duration of the game. The frequency of spawning is determined by the `spawn_interval` configuration property of that level, which determin m being initially at that given value * App.FPS, it c r, next to the display of where balls yet to be spaw he same as the configuration file (only the awn interval counter reaches 0, the next ball is spawned, will gradually move to the left

When balls spawn in the g -2, or 2 pixels in the x direction, and -2, or 2 pixe would be halved). Throughout this documen ty in the x direction and `j` is the velocity in the es which change their velocity vector trajectory,

Hitbox

A hitbox is a series of poin , a player-drawn line may appear as below:



The hitbox comprises of points (shown in red) that create line segments. V is the velocity vector of the ball, and U is the new velocity vector when it hits the line, shown in purple. N1 and N2 are the normal vectors of the line, shown in green.

The steps to calculate the new trajectory (u) could be as follows:

1. Determine the line segment that the ball is hitting (if the ball will hit any line segments). To do this, check the distance of the ball (x,y) + velocity of the ball, between two adjacent points, P1 and P2. The distance formula is as below:

$$distance(A, B) = \sqrt{(B_x - A_x)^2 + (B_y - A_y)^2}$$

   If distance(P1, ball+v) + distance(P2, ball+v) < distance(P1,P2) + ball's radius, then the ball is considered to be colliding with the line segment connecting P1 and P2.

2. Calculate the normal vectors of this line segment, N1 and N2 from P1(x1,y1) and P2(x2,y2). If we define dx = x2 - x1 and dy = y2 - y1, then the normals are (-dy, dx) and (dy, -dx).[1]

3. Normalise the normal vectors so that their magnitude is 1. (ie, divide by $\sqrt{i^2 + j^2}$).

4. Find the normal vector on the side of the line that we want to use, either N1 or N2. The one that should be used is t eck the distance of the midpoint of the li e are the blue points shown in the diag ance.

5. Calculate the new la: [1]

   Where v · n is the ctor of the line segment.

Hitboxes for player-drawn f thickness 10 units. Once a collision occurs, th

Walls

A wall is a tile with a hitbo on handling for walls will work the same as abo ts an orange, blue, green or yellow wall, it wi

Even if the game board is not surrounded by walls at the edges, balls should still reflect off the edges.

Holes

Holes take up 2x2 regular tile spaces (64x64 pixels). When a ball is within 32 pixels of the centre of a hole (from the centre of the ball), it starts to be attracted into the hole. Its size reduces proportionally to how close it comes to the centre of the hole, until when it is on top of the centre, then it will be captured by the hole and disappears. The force of attraction is approximately 0.5% of the vector from the ball to the centre of the hole.

[1] Source: https://stackoverflow.com/questions/1243614/how-do-i-calculate-the-normal-vector-of-a-line-segment

[1] Source: https://math.stackexchange.com/questions/13261/how-to-get-a-reflection-vector

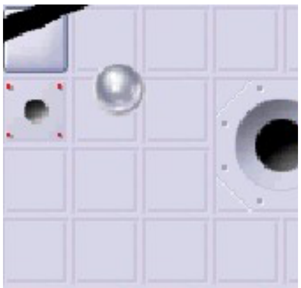


Figure 4.
Your program needs to show a proportional reduction in the ball's size, to give the illusion of it falling into the hole. To do this, specify width and height of the sprite when drawing it. You must calculate the force (ie, acceleration) of attraction by adding a proportion of the vector from the ball to the hole, to the ball's velocity on each frame.

If the hole colour matches the ball's colour (or it's a grey ball, or grey hole), it is a success and the score increases by the amount given in the configuration file, multiplied by the level multiplier. Grey balls are allowed to enter any holes, and balls of any colour can enter a grey hole to count as a success.

If the colour capture was not successful, the ball rejoins the queue of balls yet to be spawned, and score will instead decrease by the amount specified in the configuration file.

Player Actions

During the game, players

- Press 'r' to restart re completed, restart the game. The lev nd clearing any player drawn lines. Score
- Press spacebar to r is pressed again to un-pause the gam d. To indicate that the game is paused, d

Players can draw lines wit ght-clicking over them.

Score and Timer

The score value is persiste starts at the value specifie down each second. When the message === TIME'S U then press 'r' to restart th move and the player cann

Level End and Game End

A level ends when all balls are captured by holes successfully, (ie, there are no more balls remaining to be spawned, and no balls currently in play). Any remaining time gets added to the player's score, at a rate of 1 unit every 0.067 seconds. As this is occurring, two yellow tiles which begin in the top left corner and bottom right corner will move in a clockwise direction around the edge of the game board, also at a rate of 1 tile every 0.067 seconds.

When the last level ends, the game ends – display === ENDED === in the top bar.

The user may press 'r' to restart the game.