Comet Pinball

Game manual

Team Members:
Patrick Haring
Christian Bürgi

Client:
Jean-Pierre Caillot

Revision hash: 6de174c

Commit time: 2013-01-20 18:19:41 +0100

https://github.com/boskoop/comet-pinball/

Contents

List of Figures

1 Aim of the game

Pinball is a classic arcade game in which points are scored by manipulating so-called flippers in order to hit targets on a play field with a steel ball and preventing it from leaving the field. A game consists out of 3 balls which the player in turn is able to plunge into play after the previous has left the field. The game is over if the last ball has left the field and therefore the player isn't able to score points anymore.

The game consecutively adds all points up to a score. It is desirable to score as many points as possible in a game. The best games will be tracked in a high score table where the players can compare themselves with others.

2 Elements of the game

2.1 Play field

The play field is an inclined plane on which a ball will tend to roll downwards. An example play field can be seen in figure 1.

The play field has on it's right side a tube which is called a plunger tube. When bringing a new ball into play, it will be placed at the lower end of the plunger tube and can be brought into play using the plunger.

At the lower end of the field are two flippers which are used by the player to prevent the ball from passing between them into the so-called drain.

On the left side of the drain the score is displayed, on the right side the current round (ball) of play.

2.1.1 Bumpers

Bumpers are circular obstacles which react on a ball impact by applying force on the ball away from itself. Normally a player will get points for hitting a bumper as seen in figure 2a.

2.1.2 Slingshots

Slingshots are obstacles with a reactive side. If the ball hits the reactive side, it is catapulted away from the slingshot. Normally a player will get points for hitting a slingshot as seen in figure 2b.

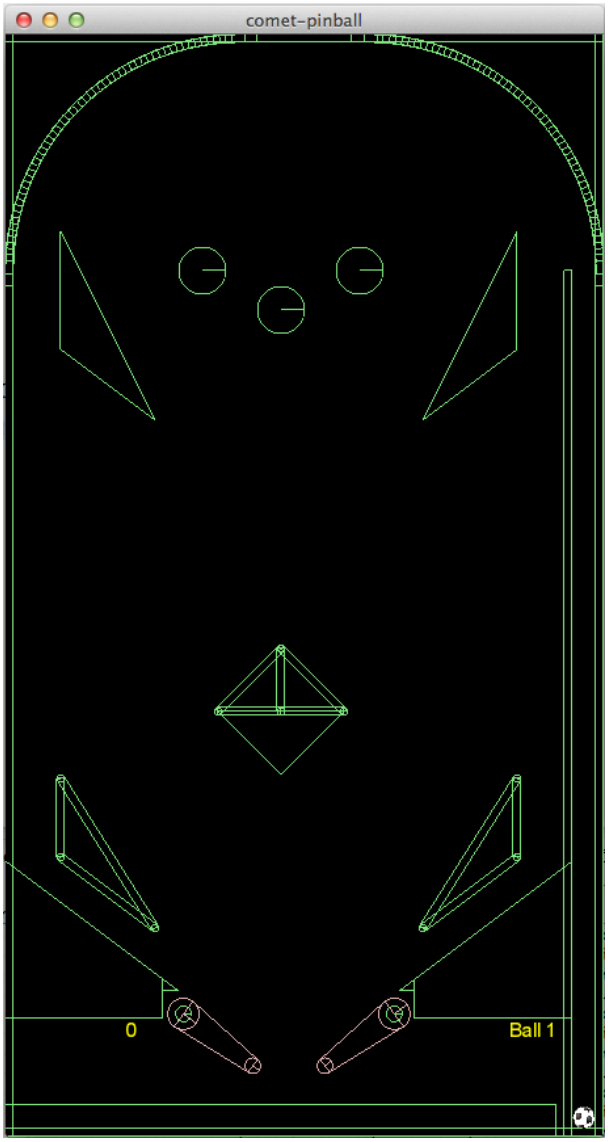


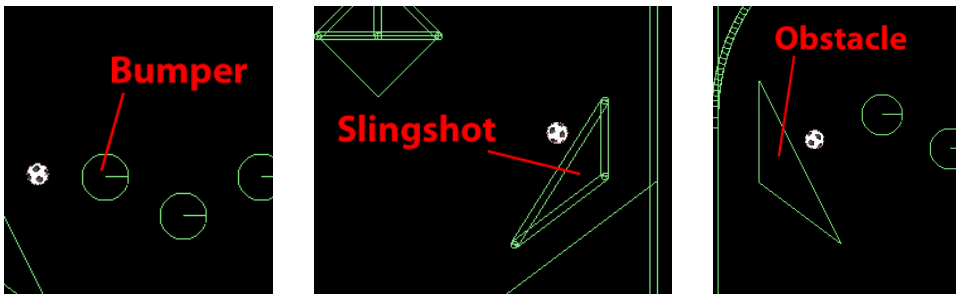
Figure 1: The pinball play field when starting a new game

2.1.3 Obstacles

Obstacles are solid bodies on the play field which generally do not react on ball impact. The deflect the ball in its trajectory and make it harder for players to hit specific elements on the play field. An obstacle can be seen in figure 2c.

2.2 Flippers

Flippers bat the ball up on the play field an prevent the ball from leaving through the drain. See figure 3.



(a) Bumper (b) Slingshot (c) Obstacle

Figure 2: Elements on a play field

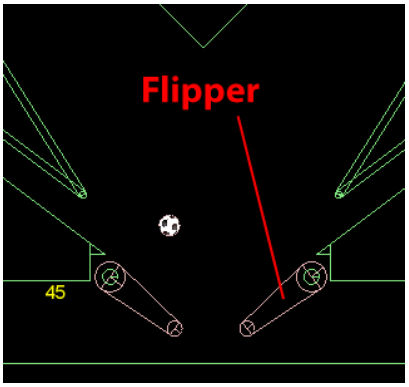


Figure 3: Flipper

3 Controls

The game comes with a default set of controls which can be customised (see section 5). The following controls are used:

Control	**Key**
Exit game	Esc
Plunge ball	Spacebar
Left flipper	Tab
Right flipper	Enter
Reset ball / End game	R

4 Visuals

4.1 Main menu

When starting the game the player will find himself in the main menu (figure 4). From there a new game can be started by pressing the button *Start Game* or the high score

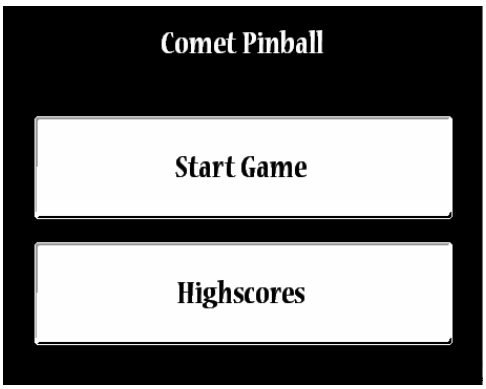


Figure 4: Main menu

can be seen by clicking on *Highscores*.

4.2 Highscores

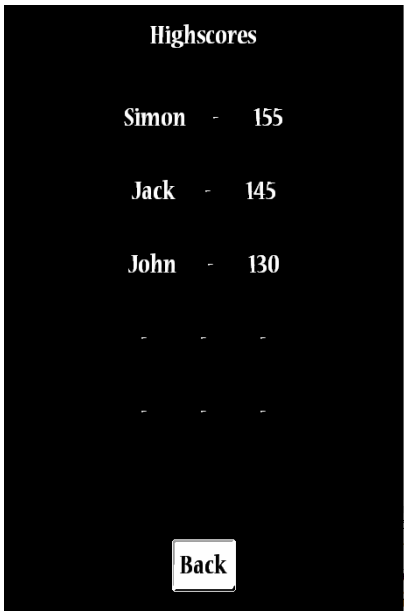


Figure 5: Highscores

On the high score screen (figure 5) the five best players are listed in descending order of their result. By clicking on *Back* the player will get back to the main menu.

4.3 Game over

After all balls have left the field the player is *game over*, which is indicated on the play field (figure 6). In order to get to the next screen, the player has to press *End game*

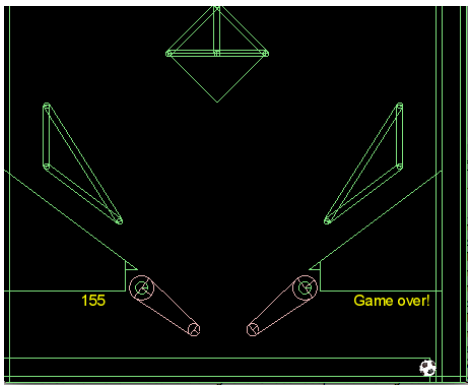


Figure 6: Game over

(default key: R).

4.4 Enter name

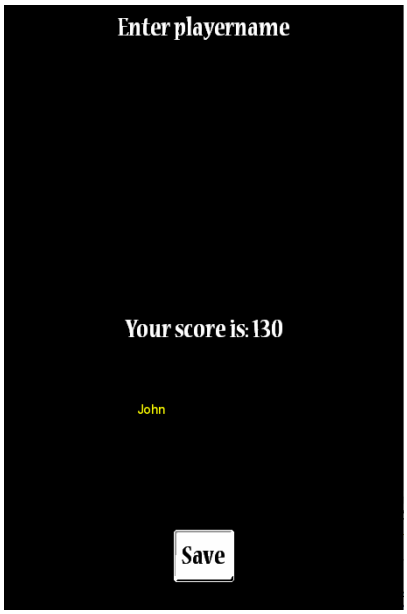


Figure 7: Player name screen

After the game has ended, the player is asked to enter his name as seen on figure 7. After having entered his name, the player should press *Save* in order to save his result and get to the high score screen.

5 Configuration

5.1 Play field

The playfield is configured through a file named *playfields.xml* in the same folder as the game jar. If the file doesn't exist on first execution, it will be created with default content.

Listing 1: playfields.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://comet.m02.ch/pinball/playfield" xmlns:xsi="
    http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://comet.m02.ch/pinball/playfield https://raw.
         github.com/boskoop/comet-pinball/master/schema/playfield.xsd">
  <playfields>
    <playfield>
      <name>default</name>
      <elements>
        <!-- elements -->
      </elements>
      <rules>
        <!-- rules -->
      </rules>
    </playfield>
  </playfields>
</configuration>
```

The configuration file contains a tag `playfields`, in which any number of tags of type `playfield` can be placed. Thereby it would be possible to support multiple play field configurations. At the moment this is not implemented and game will always load the first play field in the file.

A play field consists of three parts; the name, the elements and the rules. The name should be unique amongst all defined play fields in the file. The `elements` describe placable obstacles (`bumper`, `slingshot`, `obstacle`) and give them an id, which should be unique per play field. The rules assign game logic with play field elements. Using ids and a specific rule class they assign for example points for hitting an obstacle to a specific element.

The elements will be placed using absolute coordinates of the play field (origin is bottom-left, y points upwards, x to the right). The play field is 0.76 m * 1.40 m in size, all measures are given in meters.

5.1.1 Bumper

A bumper has an `id`, a `position` and a `radius`. The `id` should be unique amongst all elements, the `position` and `radius` are given in meters.

Listing 2: bumper

```
<configuration>
  <playfields>
    <playfield>
      <!-- name -->

      <elements>
        <bumper>
          <id>1</id>
          <position>
            <x>0.25</x>
            <y>1.1</y>
          </position>
          <radius>0.03</radius>
        </bumper>

        <!-- other elements -->
      </elements>

      <!-- rules -->
    </playfields>
  </playfield>
</configuration>
```

5.1.2 Slingshot

A slingshot has an `id`, a `position` and two corners, `corner.a` and `corner.b`. The `id` should be unique amongst all elements.

Figure 8 pictures how a slingshot is defined. The origin is given by the `position` of the element, `corner.a` and `corner.b` are defined using a vector (x/y) relative to the origin.

Important: The corners origin, a and b must always form a triangle in *anticlockwise* direction, or the element will not be simulated correctly!

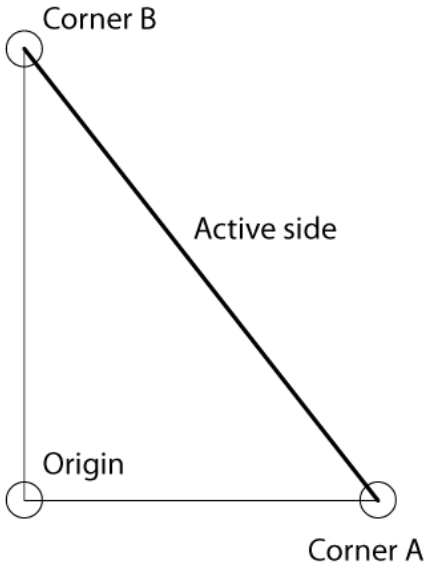


Figure 8: Slingshot element

Listing 3: slingshot

```
<configuration>
  <playfields>
    <playfield>
      <!-- name -->

      <elements>
        <slingshot>
          <id>5</id>
          <position>
            <x>0.65</x>
            <y>0.355</y>
          </position>
          <corner.a>
            <x>0.0</x>
            <y>0.1</y>
          </corner.a>
          <corner.b>
            <x>-0.12</x>
            <y>-0.09</y>
          </corner.b>
        </slingshot>

        <!-- other elements -->
      </elements>

      <!-- rules -->
    </playfields>
  </playfield>
</configuration>
```

5.1.3 Obstacle

An obstacle has an `id`, a `position` and a list of `vertices` The `id` should be unique amongst all elements.

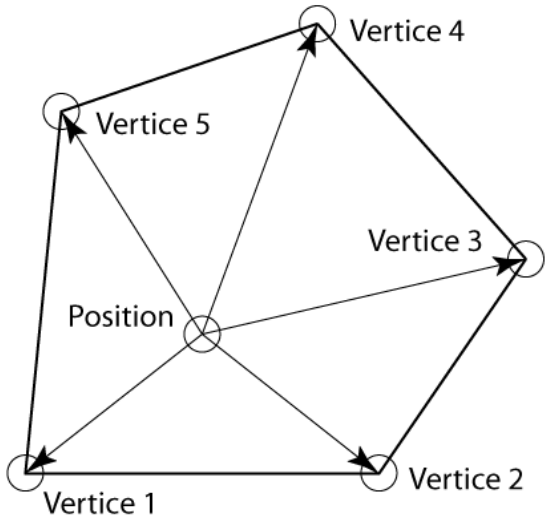


Figure 9: Obstacle element

Figure 9 pictures how an obstacle is defined. The `position` is given by absolute coordinates on the play field. Every `vertice` defines a point in relative coordinates to the `position` where a corner should be. The `position` itself can be a corner too, if there is a (0/0) `vertice`.

Important: The `vertices` must always form a *convex* polygon with corners in *anticlockwise* direction, or the element will not be simulated correctly! The number of vertices is limited to 8 per obstacle.

Listing 4: obstacle

```
<configuration>
  <playfields>
    <playfield>
      <!-- name -->

      <elements>
        <obstacle>
          <id>6</id>
          <position>
            <x>0.65</x>
            <y>1.0</y>
          </position>
          <vertices>
            <vertice>
              <x>0.0</x>
              <y>0.0</y>
            </vertice>
            <vertice>
              <x>0.0</x>
              <y>0.15</y>
            </vertice>
            <vertice>
              <x>-0.12</x>
              <y>-0.09</y>
            </vertice>
          </vertices>
        </obstacle>

        <!-- other elements -->
      </elements>

      <!-- rules -->
    </playfields>
  </playfield>
</configuration>
```

5.1.4 Rules

`Rules` define logic on a play field. A play field can have an arbitrary amount of `rules` defined. Every `rule` consists of a rule `class` and a list of `parameters`. The rule `class` must be defined using its fully qualified class name. The number of `parameters` is defined by the `rule` itself.

At the moment there is only the `rule` *HitScoreRule* implemented, which takes two parameters, an id of the affected element an the number of points the player gets, when the ball hits the element.

Listing 5: rule

```
<configuration>
  <playfields>
    <playfield>
      <!-- name -->
      <!-- elements -->

      <rules>
        <rule>
          <class>ch.m02.comet.pinball.logic.simulation.rule.basic.
              HitScoreRule</class>
          <parameters>
            <parameter>1</parameter>
            <parameter>20</parameter>
          </parameters>
        </rule>

        <!-- other rules -->
      </rules>
    </playfields>
  </playfield>
</configuration>
```

5.2 Reset stats

On first execution, the game will create a file named *pinball.ser* which contains persistent information about played games. In order to reset all statistics the file can be deleted.

5.3 Properties

In the root of the game jar, there is a configuration file *pinball.properties* located. Using these properties, the pinball game can be configured.

Listing 6: pinball.properties

```
pinball.debug=false
pinball.skip.splashscreen=false

# Keys
key.game.exit=ESCAPE
key.ball.up=UP
key.ball.left=LEFT
key.ball.right=RIGHT
key.ball.plunge=SPACE
key.ball.reset=R
key.flipper.left=TAB
key.flipper.right=ENTER

# Physics
physics.pinball.radius=0.0135
physics.earth.gravity=-9.81
physics.ramp.angle.degrees=7.0
```

5.3.1 pinball.debug

This property enables some more extensive logging and manipulation of the ball using the arrow keys.

5.3.2 pinball.skip.splashscreen

Enables/disables the splash screen.

5.3.3 key.*

Configures the keys for user interaction.

5.3.4 physics.*

Sets physics values, the units are in m, m/s^2 and degrees.