Project Specification - RoThro

Names: Chris, AnveshKrishna Pattaje, Justin Huang
Period 5

Overall Description

RoThro is a video game.

This project is a GUI-based game where the objective is to force a ball, initially located on the left side of the screen, into a hole on the opposite side of the screen. The player must use another controlling ball using the 4 arrow keys in order to exert forces on the main ball and avoid obstacles that are present at various locations on the screen and ultimately make the main ball pass through the hole to pass the level. There are three (3) levels in this game currently.

The controlling ball was implemented by using interfaces from java that “listened” to user presses on the keyboard. The collision mechanism is controlled completely by dyn4j’s classes and, it implements the same by interesting and complex calculations using the Separating Axis Theorem and other such mathematical concepts. Each level’s Obstacles are instantiated in the Main class, added to the Level (which is instantiated before the Obstacles), and then the RoThro class, which extends the SimulationFrame class of dyn4j’s library, handles the updates that occur for each frame, reflecting the changes on the screen.

The Joints for each Obstacle that has one are implemented using dyn4j’s Joint class, which has many subclasses. The specific details and operative constraints for each Joint of each Obstacle is stored in the RoThroJoint class.

An interesting feature of the game is the invisible Obstacles level. This is implemented by overriding the render method of the SimulationBody class in the Obstacle class and using a field that specifies whether the Obstacle should be visible for its particular level. When the controlling ball hits an Obstacle, its color is changed to that which was passed during its instantiation; until then, the render method ensures that the Obstacle’s color is that of the background.

Class/Interface Overview

Top Level Class

- RoThro – the main class that encapsulates the core logic of the game. Extends SimulationFrame
- Main – the command-line entrypoint for the java package. Creates the levels and Invokes RoThro

GUI classes

- Graphics2DRenderer – A class provided by dyn4j that renders all the Bodies in the world and places them in the precise locations on the screen, fills them with the desired color, and creates the desired Shapes, as specified when instantiating the Bodies.
- SimulationFrame – This is a class that is provided by dyn4j from its sample repository. It includes details of the GUI implementation. We have modified its implementation so that the size of the window can be customized. We will also likely change the various event handlers in the class in the future.
- SimulationBody – another class provided by dyn4j from its sample repository. This is the superclass for all objects in the world.
- Camera – This is yet another class provided by dyn4j in order to store the zoom and panning state of the camera and store the scaling factor that converts pixels to world units.

Major Classes

- Obstacle – extends SimulationBody. Has a constructor that creates the fixture from a Convex shape and initial positions. Sets the mass and the coefficients of friction and restitution for the obstacle. Also controls whether the Obstacle can move or not, and the type of Joint that Obstacle is part of, using a String.
- Ball – extends SimulationBody. Has a constructor that creates the fixture from a Circle.
- Avatar – extends the SimulationBody class. A representation of the controlling ball which is used by the player to move the target ball. This class uses the RoThroInputManager and RoThroKeyListener classes to implement the specific responses or changes that should be reflected on the screen based on which key was pressed by the user. It also initializes the color and size of the controlling ball.

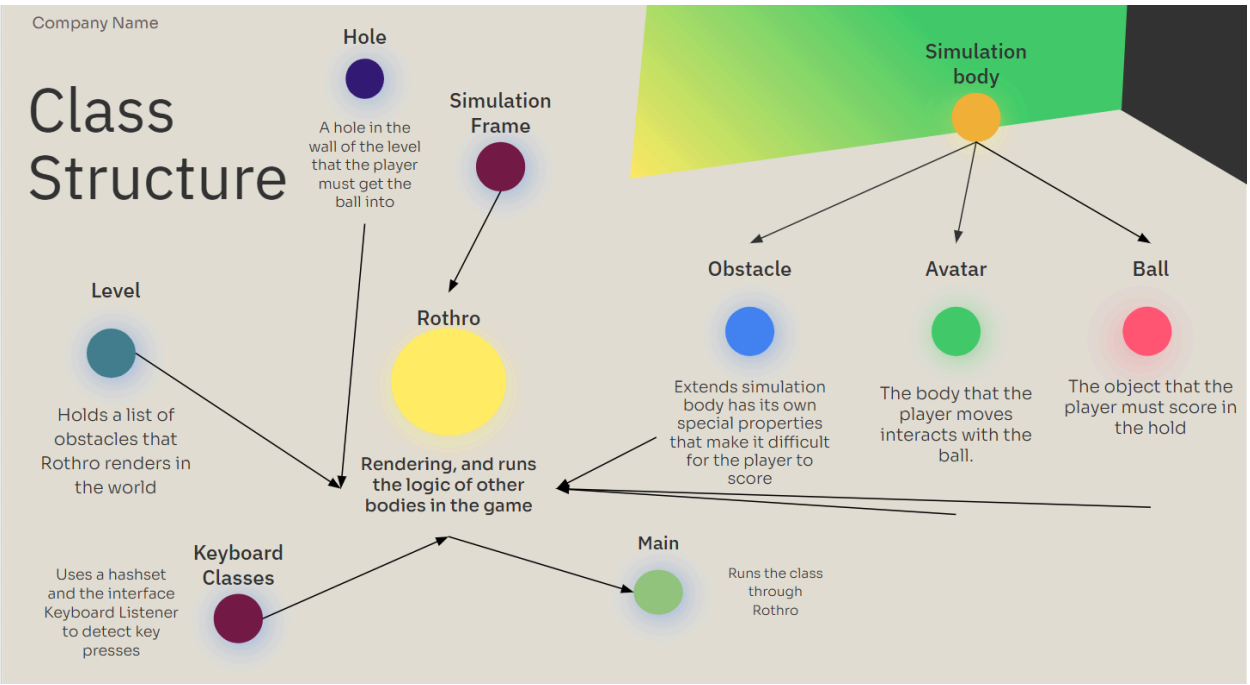
Minor Classes

- Level – Defines one level of the game. Stores the sizes and locations of obstacles in a List<List<Obstacle>>, and also stores the details and operative constraints of each Joint for each Obstacle in a List<RoThroJoint>. Also stores the radius and initial location of the main ball and the Hole for that level, which is later used in the RoThro class when starting the game.
- RoThroJoint – A class that stores the operative constraints of each Joint for the Obstacles that are involved in it and stores the Obstacle too. The implementation of the Joints becomes much more efficient because now none of the constraints for any Joint are hard-coded in the RoThro class. Instead, the RoThro class processes each RoThroJoint in each level, and based on the name, calls a

helper method. The helper method then just has to create that type of Joint while passing the constraints as specified in the RoThroJoint object.

- RoThroInputManager and RoThroKeyListener – Manages keyboard input to the RoThro game. Forwards keypresses to the other classes that take care of the underlying logic.

Rough Class Diagram



Structural Design

The following Data Structures will be used.

Description	**Data Structure**
Obstacles on the screen	LinkedList<LinkedList<Obstacle>>
Details for Joints of Obstacles	LinkedList<RoThroJoint>
The keys pressed by the player	HashSet<Integer>

Data structure rationale

It is convenient to store all the Obstacles that are currently in the world inside a LinkedList because they will not be moving around (which would require fast random

access to start them moving) and adding or removing obstacles will be faster than using an ArrayList.

The RoThroJoints, once made, will not be changed during the game. Therefore, LinkedLists are best because adding will be as easy as with an ArrayList, but removing will also be fast if required. We know we will not have to find any RoThroJoint during the game.

Each key pressed has a code associated with it. Therefore, it is best to use a HashSet. Mapping will become efficient and fast.

High Level Class Specifications

Ball

- Attributes
  - private double radius
- Methods
  - public double getRadius()

Level

- Attributes
  - private LinkedList<LinkedList<Obstacle>> obstacles
  - private ArrayList<RoThroJoint> joints
  - private Hole hole
  - private Vector2 ballPos
  - private double ballRadius
  - private int levelNo
- Methods
  - public void addObstacle(List<Obstacle> obstacle)
  - public List<List<Obstacle>> getObstacles()
  - public Hole getHole()
  - public void setBallPos(Vector2 ballPos)
  - public Vector2 getBallPos()
  - public Vector2 setBallRadius(double ballRadius)
  - public double getBallRadius()
  - public void setHole(Hole hole)
  - public boolean hasJoints()
  - public List<RoThroJointr> getJoints()
  - public void setJoints(List<RoThroJoint> joints)
  - public void addJoint(RoThroJoint joint)
  - public int getLevelNum()
  - public void setLevelNum(int levelNo)

RoThro

- Attributes
    - public static final double CAMERA_SCALE
    - public static final int WIDTH
    - public static final int HEIGHT
    - private Level level
    - private Ball ball
    - private Avatar p1
    - private RoThroKeyListener keyListener
    - public void begin(ContactCollisionData<SimulationBody> col, Contact c)
- Methods
    - protected void initializeWorld()
    - protected void initializeSettings()
    - protected void initializeCamera(Camera camera)
    - private void addPrisJoint(RoThroJoint pris)
    - private void addDistJoint(RoThroJoint dist)
    - private void addPendJoint(RoThroJoint pend)
    - private void addRevJoint(RoThroJoint rev)
    - private void addHole()
    - protected void gameLoopLogic()
    - protected void handleEvents()

Main

- Attributes
    - public static final Level[] LEVELS
    - public static Level level0()
    - public static Level level1()
    - public static Level level2()

Obstacle

- Attributes
    - private double x
    - private double y
    - private boolean movable
    - private String jointType
    - private Convex shape
    - public static final Color DEFAULT_COLOR
    - private int level
    - private String massType
    - protected boolean visible
- Methods
    - public double getX()

    - public double getY()
    - public String getJointType()
    - public Convex getShape()
    - public String getMassType()
    - public int getLevel()
    - public void setVisible(boolean visible)
    - public boolean isVisible()
    - public void render(Graphics2D g, double scale, Color color)

Hole

- Attributes
    - private double y
    - private double size
- Methods
    - public double getSize()
    - public double getY()

RoThroJoint

- Attributes
    - private String type
    - private double distance
    - private double maxMotorForce
    - private double maxMotorTorque
    - private double motorSpeed
    - private double upperLimit
    - private double lowerLimit
    - private double body1Speed
    - private double body2Speed
    - private double linearDamping
    - private double angularDamping
    - private double springFreq
    - private Obstacle body1
    - private Obstacle body2
    - private Vector2 axis
    - private Vector2 anchorPnt1
    - private Vector2 anchorPnt2
    - private boolean limitsEnabled
- Methods
    - Getters / Setters for each of the fields above

RothroKeyListener

- Attributes

    - private RoThroInputManager im
- Methods
    - public void keyPressed(KeyEvent e)
    - public void keyReleased(KeyEvent e)
    - public void keyTyped(KetEvent e)
    - public RoThroInputManager getIm()

RothroInputManager

- Attirbutes
    - private HashSet<Integer> keysPressed
- Methods
    - public void keyPressed(int i)
    - public void keyReleased(int i)
    - public HashSet<Integer> getKeysPressed()

Avatar

- Attributes
    - private ArrayList<ArrayList<Integer>> controls
    - private double r
- Methods
    - public void setControls(char option, int subjectControl, int replacementControl)
    - public void controls(HashSet<Integer> keysPressed)

RoThro.WallObstacle extends Obstacle

- Overridden Methods
    - setVisible(boolean visible)