**BreakoutFX**

<https://github.com/SecondLogic/Breakout>

04/05/2021

**Proposed Classes** |Constructors to be defined; Subject to changes during implementation

**Vector2**

+x: final double

+y: final double

+sum(vector: Vector2): Vector2

+scale(scalar: double): Vector2

+normalize(): Vector2

+dot(vector: Vector2): double

+magnitude(): double

**InputListener**

-currentScene: Scene

+getMouseLocation(): Vector2

+isPressed(key: KeyCode): boolean

+setScene(currentScene: Scene): void

**BoundingBox**

+min: final Vector2

+max: final Vector2

**SimulatedShape** extends **BoundingBox**

Enum CollisionType {BOX, CIRCLE, POLYGON}

-uiNode: Node

-collisionType: CollisionType

-position: Vector2

-velocity: Vector2

**RTree**

-root: RTreeNode

+getEnclosed(region: BoundingBox): BoundingBox[]

+getOverlapping(region: BoundingBox): BoundingBox[]

+insert(entry: BoundingBox)

+remove(entry: BoundingBox)

**RTreeNode**

-parent: RTreeNode

-children: RTreeNode[]

-isLeaf: boolean

-region: BoundingBox

+getParent: RTreeNode

+getChildren: RTreeNode[]

**ShapeCollision**

+collided: final boolean

+collisionAxis: final Vector2

+shape0: final SimulatedShape

+shape1: final SimulatedShape

**SimulationSpace**

-uiChildren: ObservableList<Node>

-staticShapes: ListProperty<SimulatedShape>

-activeShapes: ListProperty<SimulatedShape>

-rtree: RTree

-lastSimulationTick: long

+setPane(pane: Pane)): void

+simulate(): void

+resetTick(): void

**BreakoutRoom** extends **SimulationSpace**

-paddle: SimulatedShape

-ball: SimulatedShape

-bricks: ListProperty<SimulatedShape>

-turns: int

+setPaddleXPosition(xPos: double): void

**Main** extends **Application**

<uses> BreakoutRoom

<uses> InputListener

**Design Notes**

The idea behind this design is that the game can be generalized by creating a simple 2D physics engine and representing each element of the game as a physics instance. The reason for choosing this design path is to give myself more of a practical challenge as well as explore new data structures that may be relevant in my future work/studies. By designing a package that generalizes the mechanics of the game, the package can be used to create similar games without needing to write a program from scratch.

**Design Challenges**:

* Figuring out an efficient RTree implementation for this specific application
* Collision detection using RTree queries
* Collision handling between shapes (Circle – Rectangle)
* JSON encoder may be required for loading stored data into BreakoutRoom objects instead of hardcoding the levels into the program
* Menu UI is going to look like garbage as I am not a graphic designer.