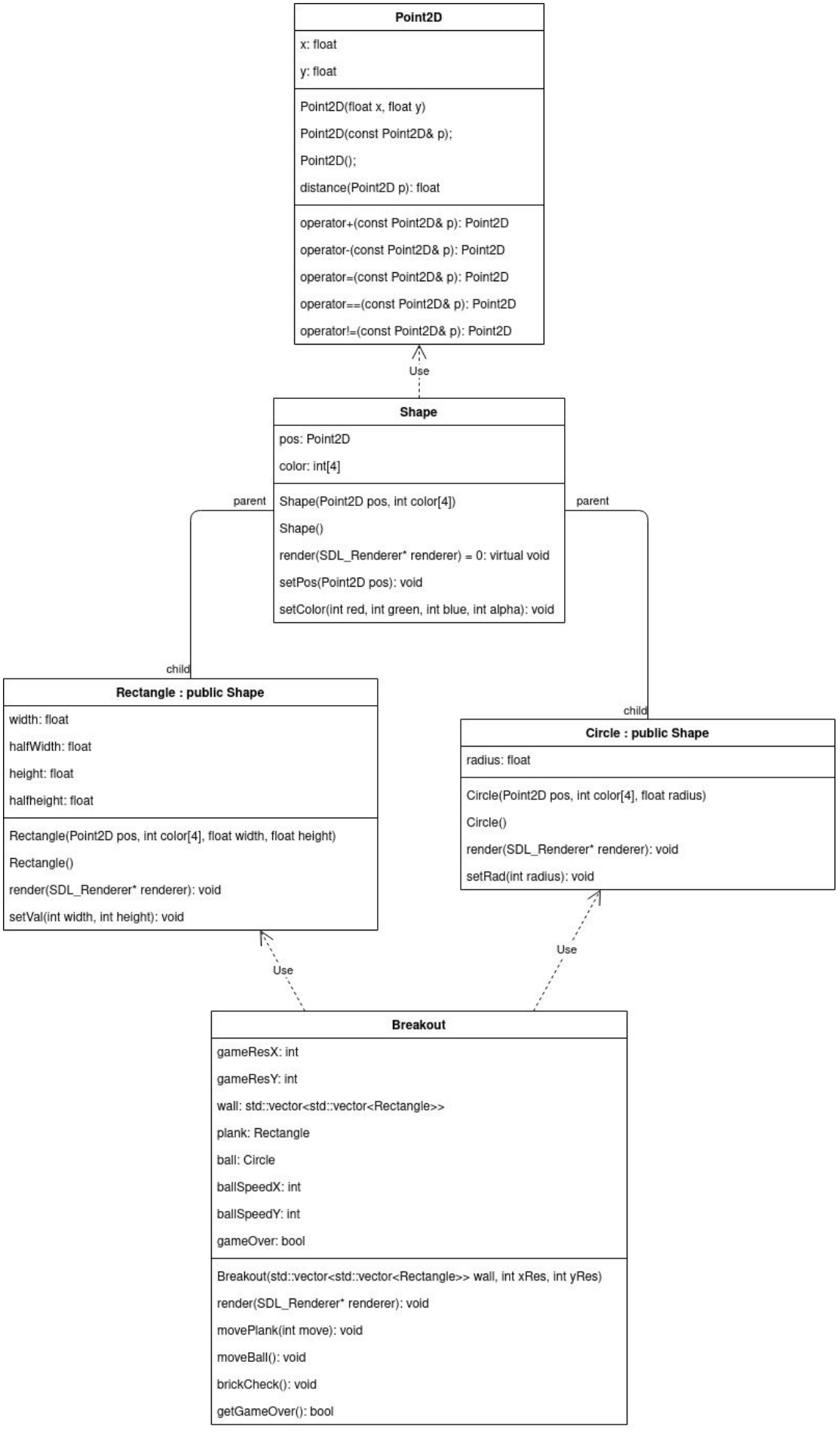
Jesper Kristensen
D0037D: Lab 6: Task 2

Breakout technical document



Jesper Kristensen
D0037D: Lab 6: Task 2

Classes and functions

class Point2D

This class holds an X and Y value and is used to calculate where all the game objects are on the screen.

Point2D(float x, float y)

Sets the X and Y values to the inserted floats.

Point2D(const Point2D& p)

Sets the X and Y values to the same floats as the inserted Point2D.

Point2D()

Sets the X and Y values to 0.

float distance(Point2D p)

Calculates and returns the distance between two Point2D's.

float getx()

Returns the X value.

float gety()

Returns the Y value.

operator+(const Point2D& p)

Used for addition between two Point2D's. Returns the answer as a Point2D.

operator-(const Point2D& p)

Used for subtraction between two Point2D's. Returns the answer as a Point2D.

operator=(const Point2D& p)

Used for copying the values from one Point2D to another Point2D.

operator==(const Point2D& p) and operator!=(const Point2D& p)

Used to check if two Point2D's has the same values. Returns true or false.

Jesper Kristensen
D0037D: Lab 6: Task 2

class Shape

Shape(Point2D pos, int color[4])

Sets the Shapes position to the inserted Point2D.
Checks that the color values are valid (val > 0 || val < 256).
If the color value is invalid it sets the color value to 0.

Shape()

Sets the position to 0,0 and all the color values to 0.

virtual void render(SDL_Renderer* renderer) = 0

A pure virtual function for the child classes to insert their specific render functionality.

void setPos(Point2D pos)

Update the position of the Shape.

void setColor(int red, int green, int blue, int alpha)

Update the shape color.

Point2D getPos()

Returns the current position.

int* getColor()

Returns the list of color values.

class Rectangle : public Shape

Rectangle(Point2D pos, int color[4], float width, float height)

Inserts the position and color values to the parent constructor.
Sets the width and height of the Rectangle to the inserted values.

Rectangle()

Calls upon the empty parent constructor and sets the width and height of the Rectangle to 0.

void render(SDL_Renderer* renderer)

Sets the render color and renders the Rectangle.

void setVal(int width, int height)

Updates the width and height to the inserted values.

Jesper Kristensen
D0037D: Lab 6: Task 2

float getWidth()

Returns the Rectangles width.

float getHeight()

Returns the Rectangles height.

class Circle : public Shape

Circle(Point2D pos, int color[4], float radius)

Inserts the position and color values to the parent constructor.
Sets the Circle's radius to the inserted value.

Circle()

Calls upon the empty parent constructor and sets the Circle’s radius to 0.

void render(SDL_Renderer* renderer)

Sets the render color and renders the Circle.

void setRad(int radius)

Updates the radius to the inserted value.

float getRad()

Returns the Circle’s radius.

class Breakout

Breakout(int bricksPerRow, int numberOfRows, int xRes, int yRes)

Saves the game resolution for scaling.
Creates a vector of vectors holding rectangles (std::vector<std::vector<Rectangle>>) making the wall look as such, wall[row][brick] where every row is a vector of Rectangles.
The constructor scales the width and height of the bricks depending on the resolution of the gamewindow.
The platform gains the same width and height as the bricks.
The ball gains a radius scaled depending on the resolution of the gamewindow.
The ball movement speed is scaled to the resolution of the gamewindow and is set.

void render(SDL_Renderer* renderer)

Calls all the game objects render functions.

Jesper Kristensen
D0037D: Lab 6: Task 2

void movePlank(int move)

Moves the platform along the x coordinates by the inserted value.

void brickCheck()

Checks if the current position of the ball would clip through any bricks.
If yes, then remove that brick from the list and update the ball movement for it to “bounce” off the brick.

void moveBall()

Calculate the new position for the ball.
Check if the new position hits the platform and update the ball movement for it to “bounce” off the platform.
Call upon the brickCheck() function.
Check if the ball hits any of the window borders and “bounce” of them.
Set gameOver to true if the ball hits the bottom of the window.
Update the ball position.

bool getGameOver()

Returns the gameOver bool.

Game render loop

1. Handle key inputs.
    a. Right arrow key: move platform to the right.
    b. Left arrow key: move platform to the left.
    c. Esc key: Exit game loop.
2. Update the ball position
    a. Check if the ball hits anything and handle “bounce”.
3. Check if Game Over
    a. Exit game loop if true.
4. Render Objects