Documentation

Neptun: ULLBQG **Name:** Rizwan Hussain

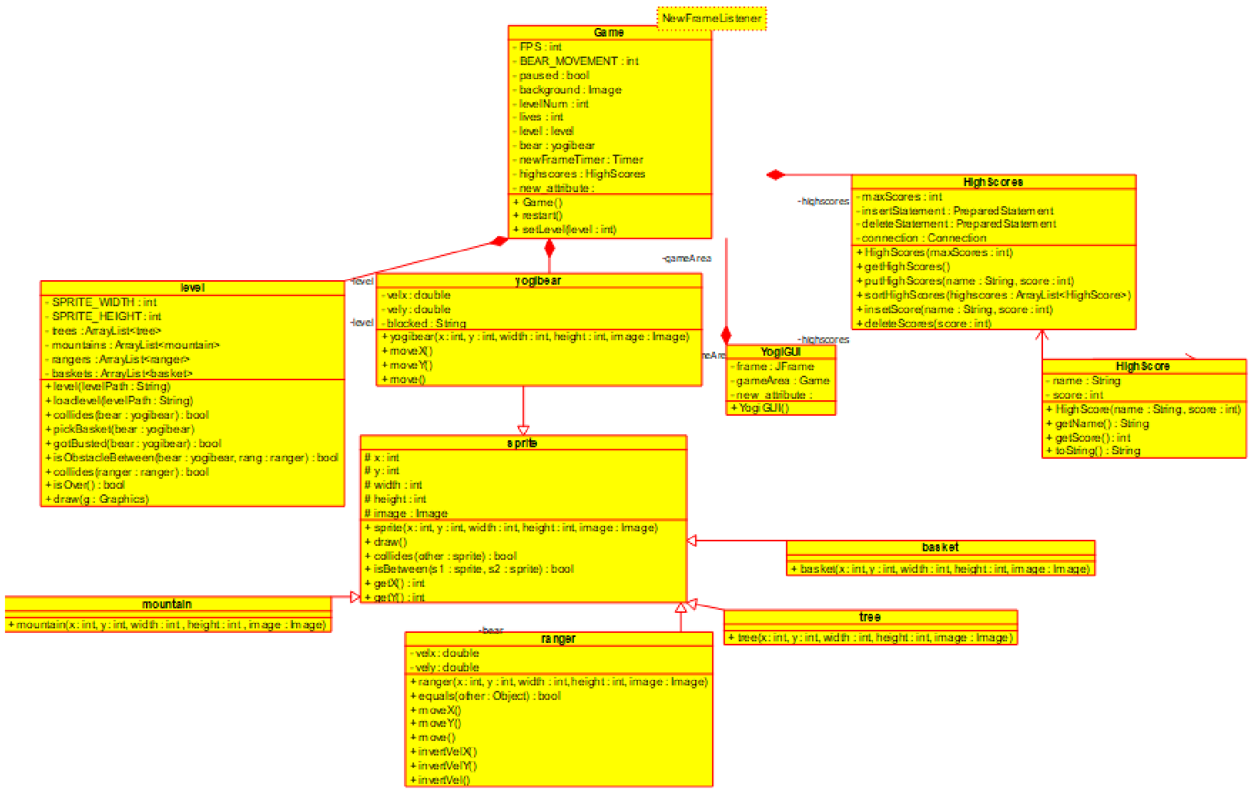
Task Description:

Yogi Bear wants to collect all the picnic baskets in the forest of the Yellowstone National Park. This park contains mountains and trees, that are obstacles for Yogi. Besided the obstacles, there are rangers, who make it harder for Yogi to collect the baskets. Rangers can move only horizontally or vertically in the park. If a ranger gets too close (one unit distance) to Yogi, then Yogi loses one life. (It is up to you to define the unit, but it should be at least that wide, as the sprite of Yogi.) If Yogi still has at least one life from the original three, then he spawns at the entrance of the park.
During the adventures of Yogi, the game counts the number of picnic baskets, that Yogi collected. If all the baskets are collected, then load a new game level, or generate one. If Yogi loses all his lives, then show a popup messagebox, where the player can type his name and save it to the database. Create a menu item, which displays a highscore table of the players for the 10 best scores. Also, create a menu item which restarts the game.

Task Analysis:

We will make a sprite object and make ranger, yogibear, tree, mountain and basket as it children. Basket, mountain and tree are immobile but yogibear and ranger are moving. Yogibear moves on pressing the arrow keys while the rangers are constantly moving without any keypress. After finishing the baskets in one level, yogi moves to the next level. Besides this, we maintain a database containing the table of the top ten scores yet scored.

UML Diagram:


NewFrameListener
Game
- FPS : int
- BEAR_MOVEMENT : int
- paused : bool
- background : Image
- levelNum : int
- lives : int
- level : level
- bear : yogibear
- newFrameTimer : Timer
- highscores : HighScores
- new_attribute :
+ Game()
+ restart()
+ setLevel(level : int)
-gameArea
-highscores
HighScores
- maxScores : int
- insertStatement : PreparedStatement
- deleteStatement : PreparedStatement
- connection : Connection
+ HighScores(maxScores : int)
+ getHighScores()
+ putHighScores(name : String, score : int)
+ sortHighScores(highscores : ArrayList<HighScore>)
+ insetScore(name : String, score : int)
+ deleteScores(score : int)
-level
-level
level
- SPRITE_WIDTH : int
- SPRITE_HEIGHT : int
- trees : ArrayList<tree>
- mountains : ArrayList<mountain>
- rangers : ArrayList<ranger>
- baskets : ArrayList<basket>
+ level(levelPath : String)
+ loadlevel(levelPath : String)
+ collides(bear : yogibear) : bool
+ pickBasket(bear : yogibear)
+ gotBusted(bear : yogibear) : bool
+ isObstacleBetween(bear : yogibear, rang : ranger) : bool
+ collides(ranger : ranger) : bool
+ isOver() : bool
+ draw(g : Graphics)
yogibear
- velx : double
- vely : double
- blocked : String
+ yogibear(x : int, y : int, width : int, height : int, image : Image)
+ moveX()
+ moveY()
+ move()
-highscores
YogiGUI
- frame : JFrame
- gameArea : Game
- new_attribute :
+ YogiGUI()
HighScore
- name : String
- score : int
+ HighScore(name : String, score : int)
+ getName() : String
+ getScore() : int
+ toString() : String
sprite
# x : int
# y : int
# width : int
# height : int
# image : Image
+ sprite(x : int, y : int, width : int, height : int, image : Image)
+ draw()
+ collides(other : sprite) : bool
+ isBetween(s1 : sprite, s2 : sprite) : bool
+ getX() : int
+ getY() : int
basket
+ basket(x : int, y : int, width : int, height : int, image : Image)
mountain
+ mountain(x : int, y : int, width : int, height : int, image : Image)
tree
+ tree(x : int, y : int, width : int, height : int, image : Image)
-bear
ranger
- velx : double
- vely : double
+ ranger(x : int, y : int, width : int, height : int, image : Image)
+ equals(other : Object) : bool
+ moveX()
+ moveY()
+ move()
+ invertVelX()
+ invertVelY()
+ invertVel()

Implementation:

Generating a level:

A level is generated from a text file. Text files are named after the number of level contained in them (00..09). Tree is the file are represented by letter 'T', mountains by letters 'M', baskets by 'B' and rangers by 'R'. We parse the file letter by letter, when we encounter any of these letters, we import respective picture from the folder and display it on the frame and add respective object to its ArrayList. The image is the size of given sprite width and sprite height. And upon encountering the empty space, we leave the space equal to the size of sprite empty.

Collision Detection:

We have introduced a method collides in sprite, which makes two rectangles of the self and the other sprite and check if they intersect. Rangers bounce back on collision with any of the immobile objects including themselves.

Live Loss:

Yogi loses its life even when it is at the distance of one unit from the rangers. To detect this beforehand, we make 8 new yogibears around the original bear and detect if they collide with any of the rangers. When this collision happens, yogibear loses one of its 3 lives. We also check if there is an obstacle between the yogi and the ranger, if so then no collision.

EventHandlers:

Arrowkeys:

Arrowkeys are used to move the yogi in the game. On pressing of a certain key, we set the respective velocity to negative or positive of the fixed velocity of yogiBear(BEAR_MOVEMENT). Like if the left key is pressed, the Xvelocity of the bear is set to the negative of bear_Movement. Before moving the bear to a new position, we detect beforehand if the new position collides with ay of the obstacles. If so, we don't move the bear. At every step, we also check if the bear steps on any of the baskets and if so, we remove a basket from the baskets list.

ExcapeKey:

When Escape key is pressed, if the game is not paused, we pause it and un-pause it otherwise.

Timehandlers:

Time is also an event in out game which happens every 1000/240 ms. Every time this event occurs, we detect whether the game is not paused, if so, we move the rangers. Then we check if the bear is busted by the rangers. If so, we decrease the lives of the bear by one and if the lives become equal to 0 we ask for the player's name and if his/her scores are better than the any of the already present scores in the table, we replace them. If lives are more than 0 then the bear is moved to the start of the level. And if the bear has picked all the baskets in the level, if the levelNum is less than 9, we load the next level, otherwise we finish because there are no more levels. And then we call the repaint.