Elad Israel 313448888

list of interfaces and classes:

Alien- describes an alien. Creates it and calculates its course of movement, and what to to when hit. Similar to block with alien background.

AlienRemover- removes Aliens from game.

AliensGroup- creates an aliens matrix(group) and implement their related methods, such as getting most left/right/up/down alien coordination in the formation, and choosing random alien from the bottom that will shoot.

Animation -describes an animation object- any animation should specify what to do in each frame, and notify when to stop the animation.

AnimationRunner- The AnimationRunner takes an Animation object and runs it.

Ass7Game- main class- Creates a new arkanoid game, initialize and runs the game.

Background- The Background interface. describes a Background of a level or block.

BackgroundImage -Background with Image. describes a Background of a level or block with image.

Ball- a Ball (actually, a circle) has size (radius), color, and location (a Point). Balls also know how to draw themselves on a DrawSurface. Used as shots

BallRemover- BallRemover is in charge of removing balls from the gameLevel, as well as keeping count of the number of balls that remain.

Block- Blocks are obstacles on the screen. a Block (actually, a Rectangle) has size (as a rectangle), color, and location (a Point). Blocks also know how to draw themselves on a DrawSurface. A block can also notify the object that we collided with it about the new velocity it should have after collision.
here the blocks are used to create shields for the player.

BlockRemover- a BlockRemover is in charge of removing blocks from the gameLevel, as well as keeping count of the number of blocks that remain.

Collidable- The Collidable interface is used by things that can be collided with. A collidable object must have location and size(collision rectangle) and need to know what to do when a collision occurs.
here the collidable objects are paddle ,aliens and blocks(shields).

CollisionInfo -CollisionInfo Contains information about a collision- which object collided with any of the collidables in this collection, and the closest collision point that is going to occur.

CountdownAnimation-The CountdownAnimation will display the given gameScreen, for numOfSeconds seconds, and on top of them it will show a countdown from countFrom back to 1, where each number will appear on the screen for (numOfSeconds / countFrom) secods, before it is replaced with the next one.

Counter- Counter is a simple class that is used for counting things.
here it is used to count the score, game level number, etc..

EndScreen- Once the game is over (either the player run out of lives or managed to clear all the levels), we will display the final score. If the game ended with the player losing all his lives, the end screen should display the message "Game Over. Your score is X" (X being the final score). If the game ended by clearing all the levels, the screen should display "You Win! Your score is X". The "end screen" should persist until the space key is pressed. After the space key is pressed, the program should terminate.

GameEnvironment - A collection of the objects a Ball(shot) can collide with. The ball will know the game environment, and will use it to check for collisions and direct its movement.

GameFlow- In charge of creating the different levels, and moving from one level to the next.

GameLevel- A class that will hold the sprites and the collidables, and will be in charge of the animation.

HighScoresAnimation- the animation of the highscores.

HighScoresTable- a table representing the highscores. Will be presented at the end of game or on demand using showHighscoreTask.

HitListener- HitListener Objects that want to be notified of hit events, should implement the HitListener interface, and register themselves with a HitNotifier object using its addHitListener method.

HitNotifier- The HitNotifier interface indicate that objects that implement it send notifications when they are being hit.

KeyPressStoppableAnimation-wrap an existing animation and add a "waiting-for-key" behavior to it.

Line- A line (actually a line-segment) connects two points - a start point and an end point. Lines have lengths, and may intersect with other lines. It can also tell if it is the same as another line segment.

LivesIndicator- sprite that will sit at the top of the screen and indicate the number of lives.

Menu-When the game starts, the user will see a screen stating the game name (Arkanoid), and a list of several options of what to do next.

MenuAnimation- Our Menu will need to be displayed on screen, so it will be an Animation. Unlike the other animation loops, this one will need to return a value when it is done. We may want to add a nice background to our menu. For this, we will provide it with a method that will accept a background sprite and display it.

NameOfLevelIndicator- sprite that will sit at the top of the screen and indicate the name of the level.

Paddle- The Paddle is the player in the game. It is a rectangle that is controlled by the arrow keys, and moves according to the player key presses. It implements the Sprite and the Collidable interfaces. It also knows how to move to the left and to the right.

PauseScreen -Display a screen with the message paused -- press space to continue until a key is pressed. An option to pause the game when pressing the p key.

Point- A point has an x and a y value, and can measure the distance to other points, and if its is equal to another point.

Rectangle- A Rectangle has size, color, and location (a Point). it also has edges and fill and draw colors. Rectangles also know how to draw themselves on a DrawSurface.

ScoreIndicator- sprite that will sit at the top of the screen and indicate the score.

ScoreInfo- Contains the information about the score.

ScoreTrackingListener- ScoreTrackingListener updates the score counter when blocks are being hit and removed.

ShowHiScoresTask- a task showing the highscore table.

SpaceBackground- background class for Alien level.

Elad Israel 313448888

Sprite-A Sprite is a game object that can be drawn to the screen (and which is not just a background image). Sprites can be drawn on the screen, and can be notified that time has passed (so that they know to change their position / shape / appearance / etc)

SpriteCollection-a SpriteCollection will hold a collection of sprites.

Task-a task to run. For example ShowHiScoresTask is a task that runs in the menu.

Velocity- Velocity specifies the change in position on the `x` and the `y` axes.

הסבר קצר על:

מימוש קבוצת החייזרים:

יצרתי class המגדיר חייזר, הדומה מאוד לבלוק. עשיתי קבוצת חייזרים היוצרת מטריצה של חייזרים.
כמו כן class זה יודע להחזיר ערכי x או y של קצות המטריצה לפי מיקומי החייזרים הקיצוניים- וכך מימשתי את תזוזת החייזרים בהגיעם לקצוות המסך- אם החייזר הכי ימני הגיע לקצה הימני של המסך, תוריד את כל המטריצה שורה וכו'..

מימוש המגנים:

המגנים הם פשוט קבוצה גדולה של בלוקים(30 בלוקים בשורה, 3 שורות, 3 בלוקים כאלה סה"כ).

יצרתי אותם כמו את שאר הדברים בשלב- בGameLevel.

יריות החייזרים:

ירייה של חייזר זה פשוט כדור היוצא מתחתית החייזר וזז למטה.
את בחירת החייזר שאמור לירות ביצעתי בעזרת פונקציה בAlienGroup הבודקת מי אמור לירות בהתאם לחייזר התחתון בכל עמודה(קיימת) במטריצה.
תזמון הירייה נעשה באמצעות משתנה המתחיל ב0.5 (הזמן שבו החייזר אמור לירות :כל 0.5 שניות), וכל ריצה של doOneFrame בGameLevel מוריד ממנו dt. כשמגיעים ל0 זה אומר שצריך ליצור ירייה חדשה ואני קורא לפונקציה המתאימה.

יריות השחקן:

ירייה של שחקן זה פשוט כדור היוצא מראש הפאדל וזז למעלה.
תזמון הירייה נעשה באמצעות משתנה המתחיל ב0.35 (הזמן המינימלי שבו השחקן יכול לירות לאחר ירייה קודמת), וכל ריצה של doOneFrame בGameLevel מוריד ממנו dt. כשמגיעים ל0 זה אומר שצריך ליצור ירייה חדשה ואני קורא לפונקציה המתאימה.