DADS CRC CARDS

Legend:

BLUE = Functionality Classes

Yellow = Weapon Object Classes

Green = Friendly Object Classes

Red = Enemy Object Classes

MissileCommand.java

- Main logic loop of the game
- Keep score of the game
- Collision detection (Detects when hornets touch missile explosion or cows, calls appropriate methods)
- Start new game
- Change rounds/levels
- Draws
  - Draws the initial level screen (Foreground and Background)
  - Also calls other draw functions for all objects on screen, ie hornets, cows, etc
- Tracks FPS for animations

Interacts with:
- Base.java
- Cows.java
- Hornets.java
- Missile.java

MainActivity.java

Variables:
- MissileCommand

Methods:
- createGame()
- pauseGame()
- resumeGame()

Explanation:
MainActivity.java is responsible for creating a new instance of the game, then possibly pausing and resuming it, time allowing.

Interacts with:
MissileCommand.java to start the game.

Hornet.java

Variables:
- xVelocity
- yVelocity
- Image img
- Spawn Coords (X and Y)
- Destination Coords (X and Y)
- Position (RectF)
  - Width and Height
- isAlive

Methods:
- getPosition()
- kill()
- increaseSpeed()
- draw()
- update()

Explanation:
Hornet.java is the class that tracks the hornet stingers coming down from the sky, in place for enemy missiles in the classic game. It draws itself, kills itself when interacts with missiles or cows, and increases in speed each round.

Interacts with:
None
(other classes interact with Hornet, but not visa versa)

Base.java

Variables:
- Image img
- xPosition
- yPosition
- ammoCount

Methods:
- getPosition()
- kill()
- draw()
- fire()
- addAmmo()

Explanation:
Base.java is the class that handles the player's DADS base. The main job of Base is to create instances of Missiles that head towards the spot where the player presses the screen, and keep track of how many Missiles they have left. It also draws itself and stores its own coordinates. Finally, it also has a function to add ammo via powerups.

Interacts with:
Missile.java when fire() is called, passes origin and target coordinates and creates new instance of missile.

Laser.java when the player gets the powerup.

Building.java

Variables:
- xPosition
- yPosition
- isAlive
- Building Image

Methods:
- getPosition()
- kill()
- revive()
- draw()

Explanation:
Hornet.java is the class that tracks the Davis buildings that the DADS must defend! It is essentially the same as Cow.java, but with a different image and it cannot moo().

Interacts with:
None
(other classes interact with Building, but not visa versa)

Cow.java

Variables:
- xPosition
- yPosition
- isAlive
- Cow Image

Methods:
- getPosition()
- kill()
- revive()
- draw()
- moo()

Explanation:
Hornet.java is the class that tracks the cows that the DADS must defend! The class is very simple, as the cows only have to keep track of their position, whether they have been destroyed or not, the ability to destroy them and the ability to revive them after reaching a score threshold. Also, if the player taps on them, we plan on making them moo, time allowing.

Interacts with:
None
(other classes interact with Cow, but not visa versa)

Powerup.java

Variables:
- Image img
- xPosition
- yPosition
- yVelocity
- String powerUpType

Methods:
- getPosition()
- kill()
- draw()
- addMissiles

Explanation:
Base.java is the class that handles the player's DADS base. The main job of Base is to create instances of Missiles that head towards the spot where the player presses the screen, and keep track of how many Missiles they have left. It also draws itself and stores its own coordinates.

Interacts with:
Base.java to call its addAmmo() function.

Missile.java

Variables:
- xVelocity
- yVelocity
- xPosition
- yPosition
- xTarget
- yTarget
- missileWidth
- missileHeight
- Image img
- Explosion img
- Circle hitbox

Methods:
- getPosition()
- update()
- kill()
- status()
- draw()
- explode()
- explosionSize()

Explanation:
Missile.java is the class that tracks the player fired missiles that explode when they reach their target position after traveling at a set velocity. The explosion changes in size over time, which explosionSize() calculates. It is also resOnce the explosion ends, it also then removes itself from the screen.

Interacts with:
None
(other classes interact with Missile, but not visa versa)

Laser.java

Variables:
- RectF hitbox
- Int timer
- Color color
- xStart
- yStart
- Int width
- Int length

Methods:
- getPosition()
- draw()

Explanation:
Laser.java will replace Missile.java as the DADS weapon for a small set amount of time when a powerup contains a laser powerup instead of ammo. It allows the DADS to shoot a continuous stream that instantly explodes enemy hornets.

Interacts with:
None