DADS CRC CARDS

Legend:

BLUE = Functionality Classes Yellow = Weapon Object Classes

Green = Friendly Object Classes Red = Enemy Object Classes

BaseCtrl.java

Responsibilities:	Collaborations:
- It will draw the Base - It will access the missiles arraylist inside of Base.java - Calls the update function for all missiles in the arraylist - Draws the missiles in the arraylist, either as explosions or missiles in travel.	Base.java

Base.java

Responsibilities:	Collaborations:
- Base is where the missiles are fired from - Holds the missiles in an ArrayList - Keeps track of what type of missiles it has and spawns the correct type - Removes the missiles once it gets to its location 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.	Missile.java when fire() is called, passes origin and target coordinates and creates new instance of missile.

PowerUp.java

Responsibilities:	Collaborations:
- Keep track of what type of power up it is (laser, extra ammo, etc) - Fall down from the sky along the Y-coordinate and constant X-coordinate at a constant velocity - Keep track of it's position and velocity	The game loop checks if user's missile explosion intersects with the power up and then the user's Base.java is granted a new power up, ie increase Base.ammo.

PowerUpCtrl.java

Responsibilities:	Collaborations:
- Creates a list of PowerUps - Spawns new power ups and draws these power ups on the screen - Will also handle removing power ups from the screen	PowerUp.java

MainActivity.java

Responsibilities:	Collaborations:
- Start the game when the app is opened - Pause the game when game - Resume paused game	MissileCommand.java to start the game.

MissileCommand.java

Responsibilities:	Collaborations:
- Main logic loop of the game - Keep score of the game - Responsible for constructing all other objects, hornets, cows, powerups, etc. - Stores hornets, cows, powerups in lists - 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 - Keeps track of levels of the game - Play music for menus and starting game	- BaseCtrl.java - CowsCtrl.java - HornetCtrl.java - Missile.java - PowerUpCtrl.java - LevelCtrl.java - MainMenu.java - Background.java - Pause.java - Sound.java

Missile.java

Responsibilities:	Collaborations:
- Travel in a straight line with constant velocity from the base to user's tapped coordinate location, explode once tapped location is reached. - Store it's location and shape - Store it's explosion radius - Play launching and exploding sound effects	Game loop reads in user's OnTouchEvent and calls Base.fire() function which calls Missile.java constructor. It's instances are accessed by the game loop to check for collisions

CowsCtrl.java

Responsibilities:	Collaborations:
- Creates the Cows - Stores the Cows in an array - Draws each of the cows on the screen - Returns the current amount of cows that are alive for use in determining score at end of each level, and when the player gets a gameover	Cows.java

LevelCtrl.java

Responsibilities:	Collaborations:
- Handles the levels of the game - Keeps track of number of Hornets per level - Keeps track of number of missiles per level - Helps with the calculation of the speed of the hornets per level - Keeps track of the number of power-ups that can spawn per level	MissileCommand.java

Background.java

Responsibilities:	Collaborations:
- Draws the main canvas and the background of the game	Uses BitMap

HornetCtrl.java

Responsibilities:	Collaborations:
- Creates a list of Hornets - Spawns new Hornets and draws these Hornets on the screen - Will also handle killing and removing hornets from the screen	Hornet.java

MainMenu.java

Responsibilities:	Collaborations:
- Handles the MainMenu of the game from which we switch into the game state	- Context - Bitmap

Hornet.java

Responsibilities:	Collaborations:
- Spawn new hornets when prompted to by MissileCommand.java - Store and update its coordinate location - Destroy itself once it gets to its final destination (cow location) - Increase its velocity when certain level thresholds are reached 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 flies towards the cows in a straight line. It draws itself, kills itself when interacts with missiles or cows, and increases in speed each round.	Hornet's information is accessed by the game loop to check for collision between either a cow or a missile explosion. Hornet calls Cow.kill() when it collides with it.

Sound.java

Responsibilities:	Collaborations:
- Plays and Pauses all of the sound effects and music in the game - Uses MediaPlayers - Contains all the sound files as local variable MediaPlayers - Changes background music at a given level to match level of game intensity	- LevelCtrl.java to determine which background music to play

Cow.java

Responsibilities:	Collaborations:
- Cows are spawned at the start of the game or on each new level - Stores its location - Store its state (alive or dead)	If a stinger reaches a cow it is destroyed.

Pause.java

Responsibilities:	Collaborations:
- Stores the image of the pause button used to determine if the player paused the game or not - Calls the options class to draw the pause menu while paused	- Options.java

Options.java

Responsibilities:	Collaborations:
- Stores the images for audio on/off and restart for the pause menu - Determines if the player clicked on one of the buttons - Returns an int to MissileCommand that will act on, turning audio on/off, restarting the game, or unpausing - Flips between audio on/off pictures	- None