CSC 392/492: Mobile Applications Development for Android II

Assignment 4 – Defense Commander App *(400 pts)*

Uses: Dynamic Views, Object Animators, ValueAnimator, Interpolators, SoundPool, MotionEvents, Scrolling Backgrounds, TouchEvents, Remote Database, Threads

1) General Description:

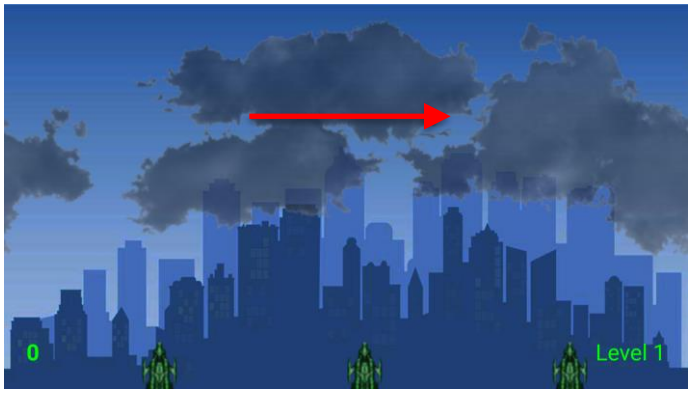
- This app is a game, a modern take on an 80's era video game where the user defends their bases against missile attacks.
- Major entities in the app include bases, incoming ballistic missiles, and interceptors
- The game increases in difficulty as time passes (i.e., Levels)
- Incoming missiles appear from the top of the screen and head towards the bottom where they will detonate, potentially destroying your bases
- You launch interceptors to destroy the incoming missiles in flight before they hit your bases
- When your bases are gone, the game is over
- Interceptors are launched by a finger-tap
- Sound effects enhance the gaming experience
- A remote database is used to track the top 10 scores among all players
- Scoring in the top 10 allows the user to enter their initials
- Scores are displayed at the end of the game

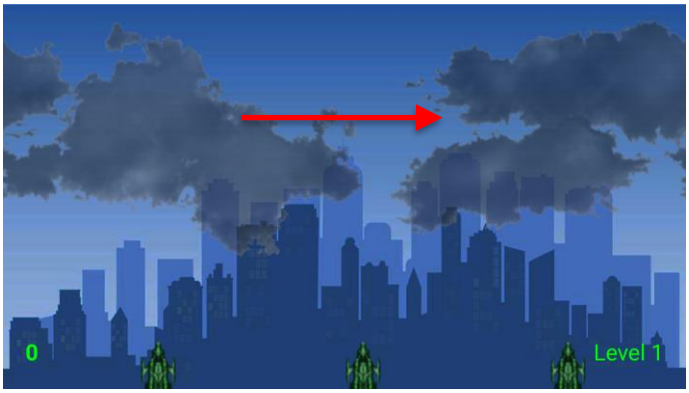
NO LATE SUBMISSIONS ARE ALLOWED FOR THIS ASSIGNMENT

2) App Behavior Diagrams

a) Splash Screen *(Title fades in, Background music starts)*

b) Game Screen at Startup (the clouds are moving from left to right)

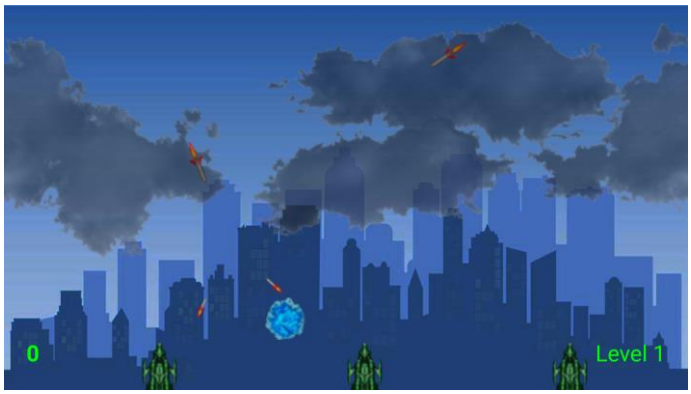




c) Inbound Missiles! *(missile sound plays as they enter the screen)*

d) Missiles Hit a Base *(base destruction sound plays when a base is destroyed, inbound missile detonation results in fiery blast, base destruction adds an additional fiery explosion)*

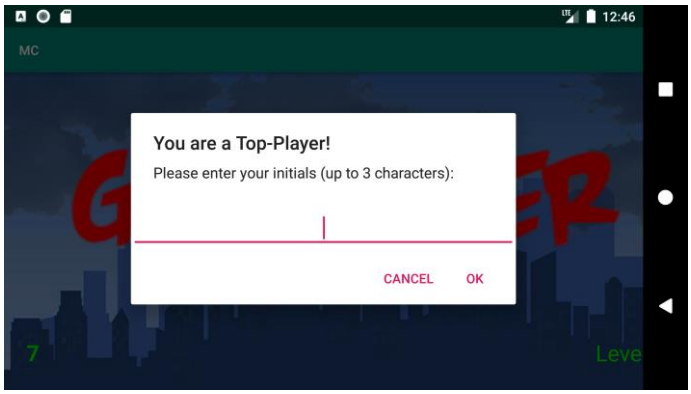
e) Interceptors *(launched by touching the screen at the desired detonation location, launch sound plays, detonation sound plays when they explode, a blue explosion appears when they detonate)*



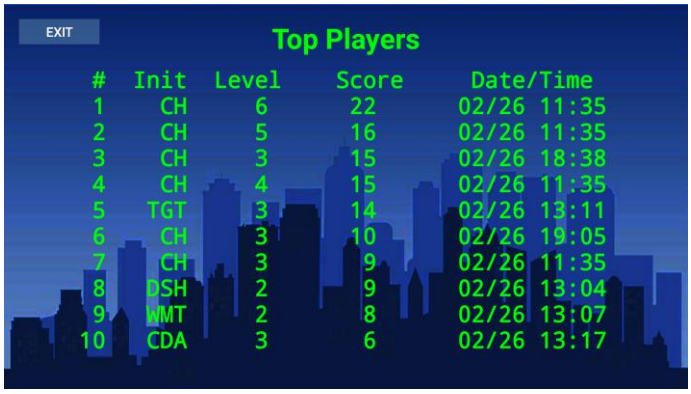
f) Intercepting a Missile *(if the interceptor explodes near the missile, the missile explodes mid-air, both play a detonation sound and show a fiery blast, 1 point is earned for each missile destroyed before it reaches the ground)*

g) No More Bases *(Title Fades in, shows for 3 seconds, then checks the score against the high scores)*

h) If the score is within the top 10 high scores, the user can enter their initials and the score is recorded on a remote database.

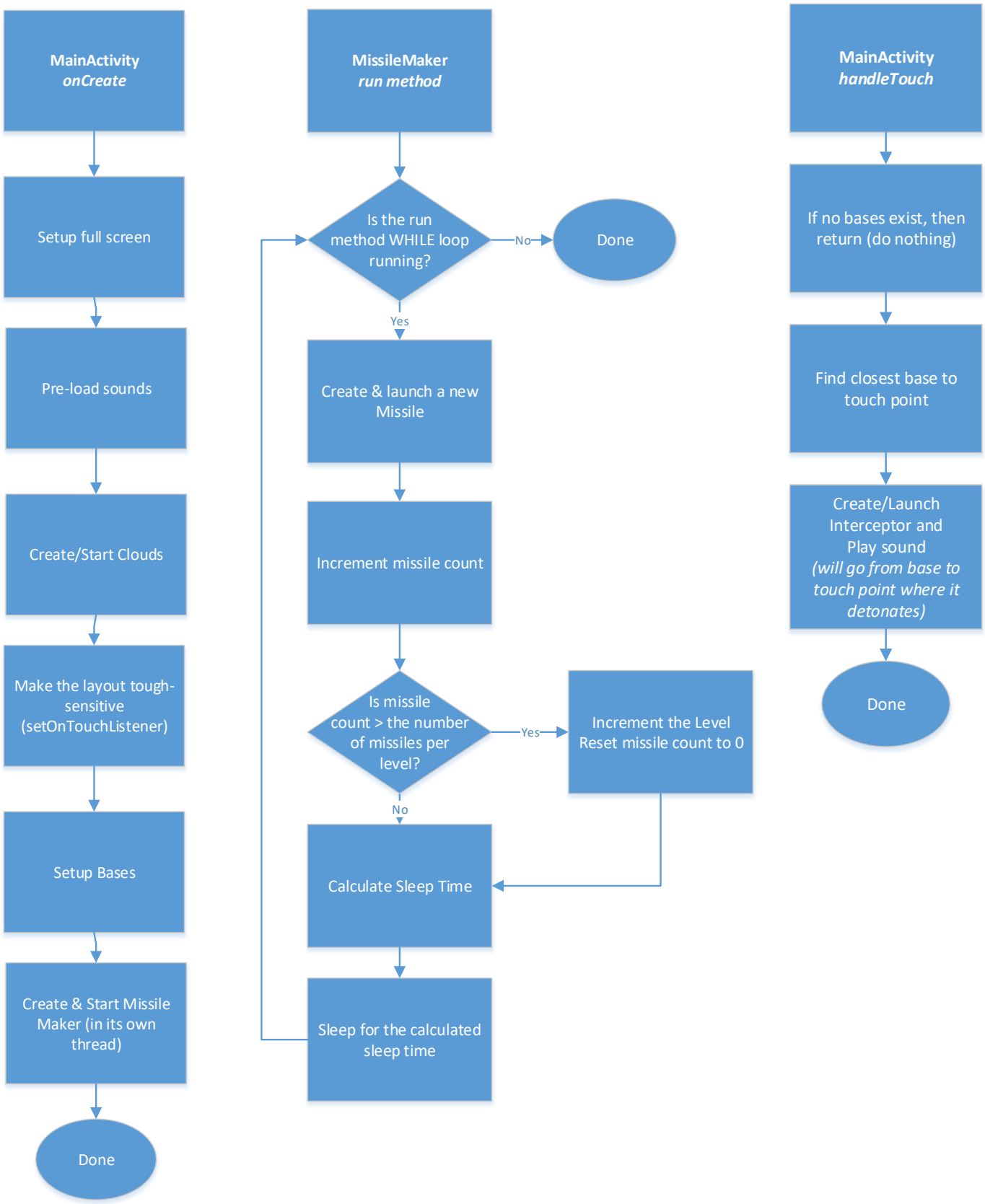


i) Top 10 scores are displayed at the end of every game

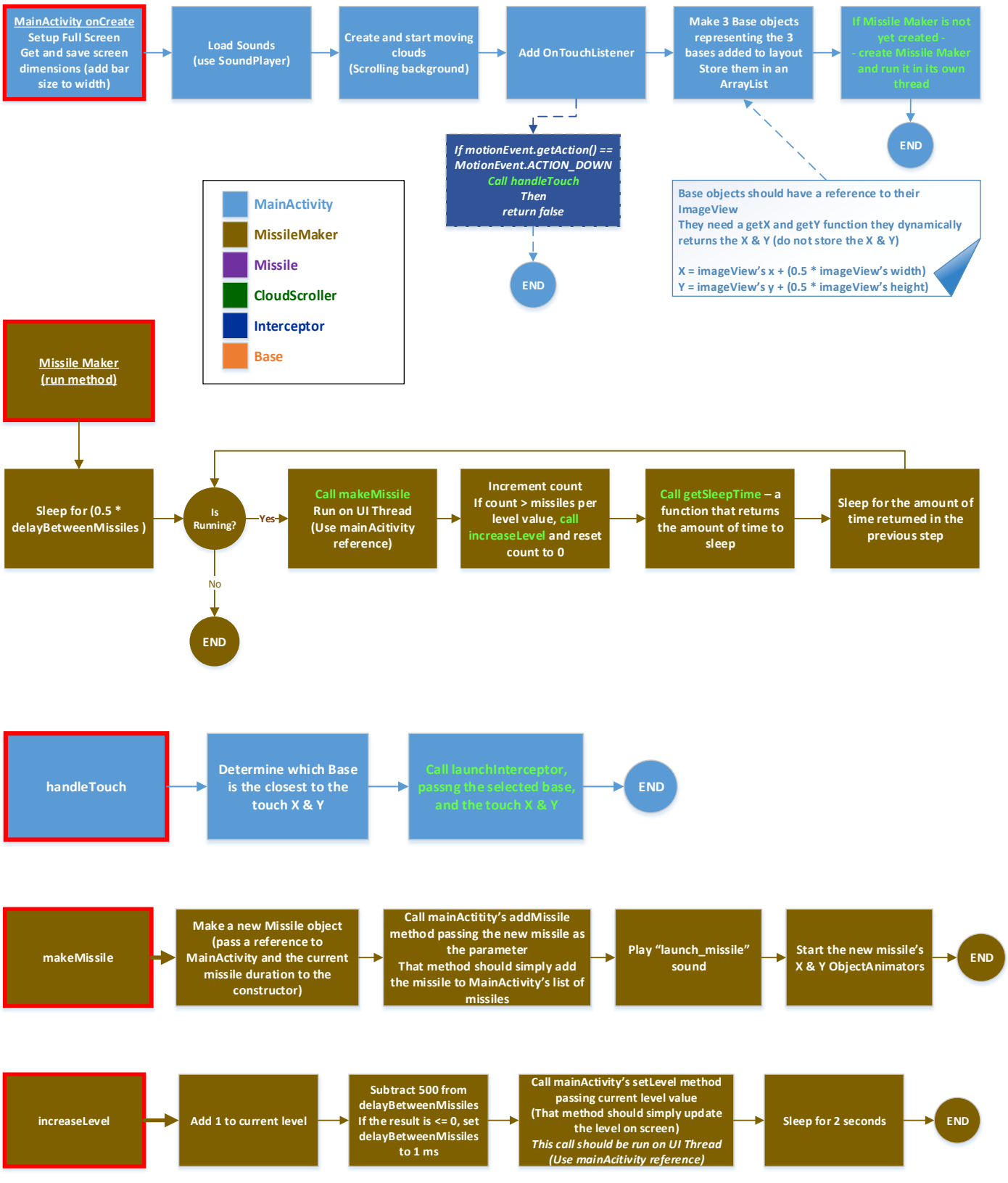


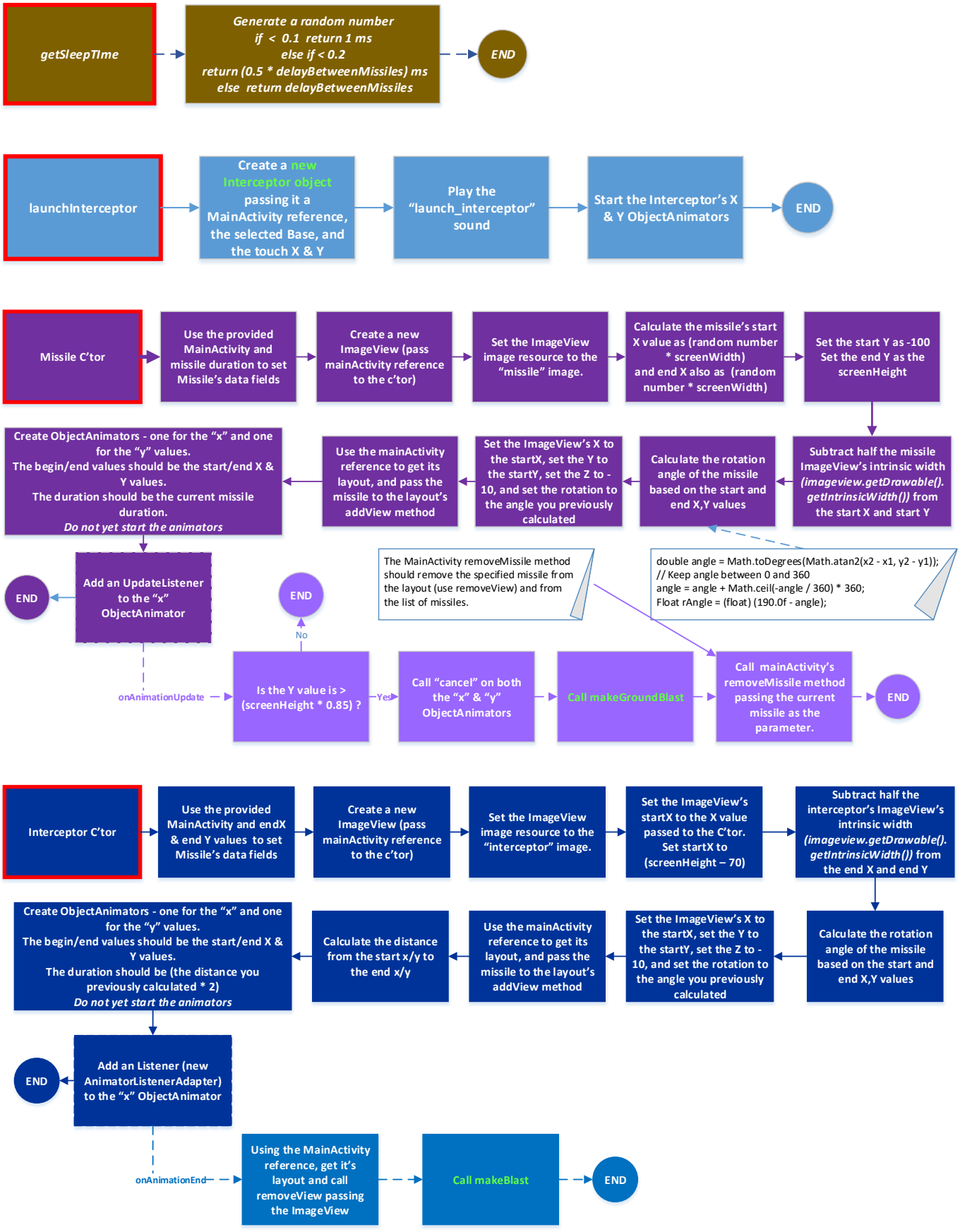
NO LATE SUBMISSIONS ARE ALLOWED FOR THIS ASSIGNMENT

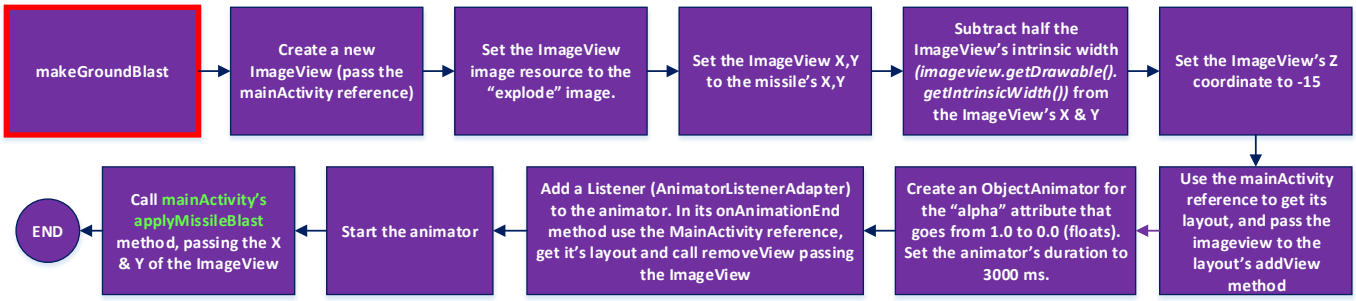
3) **High-level Behavior Diagrams** *(for selected behaviors)*

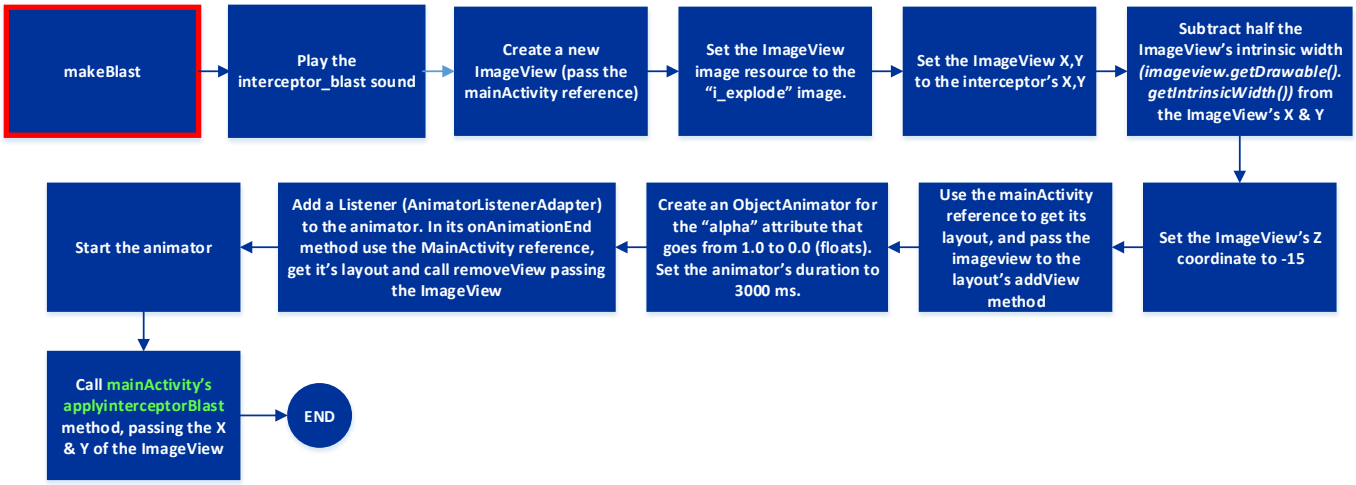


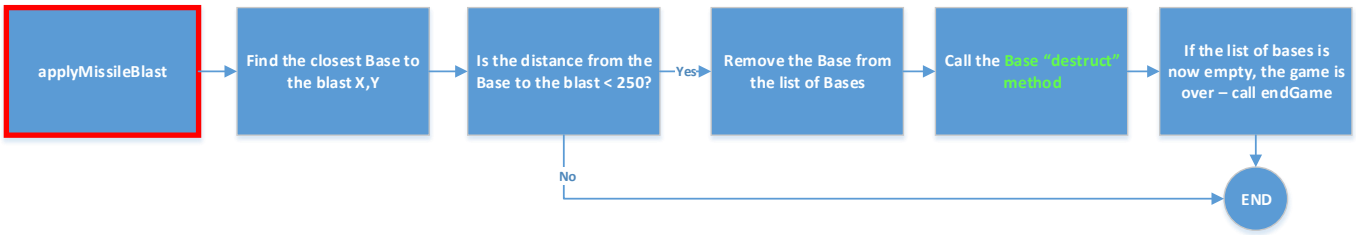
4) Detailed Behavior Diagrams

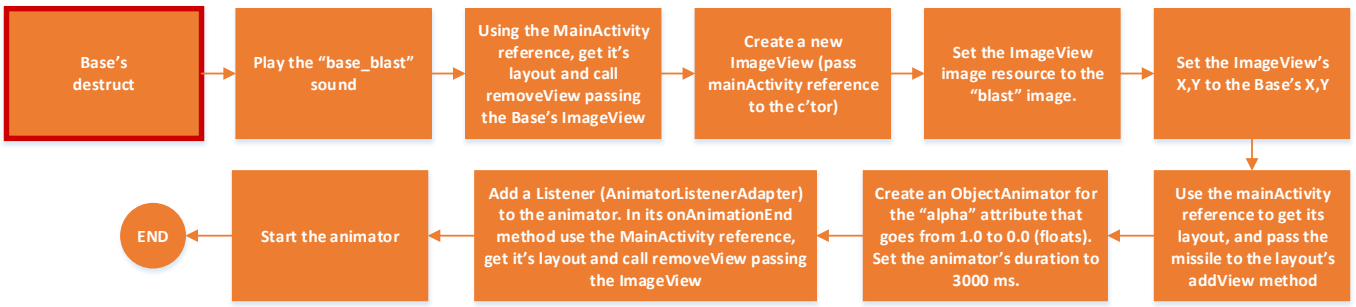


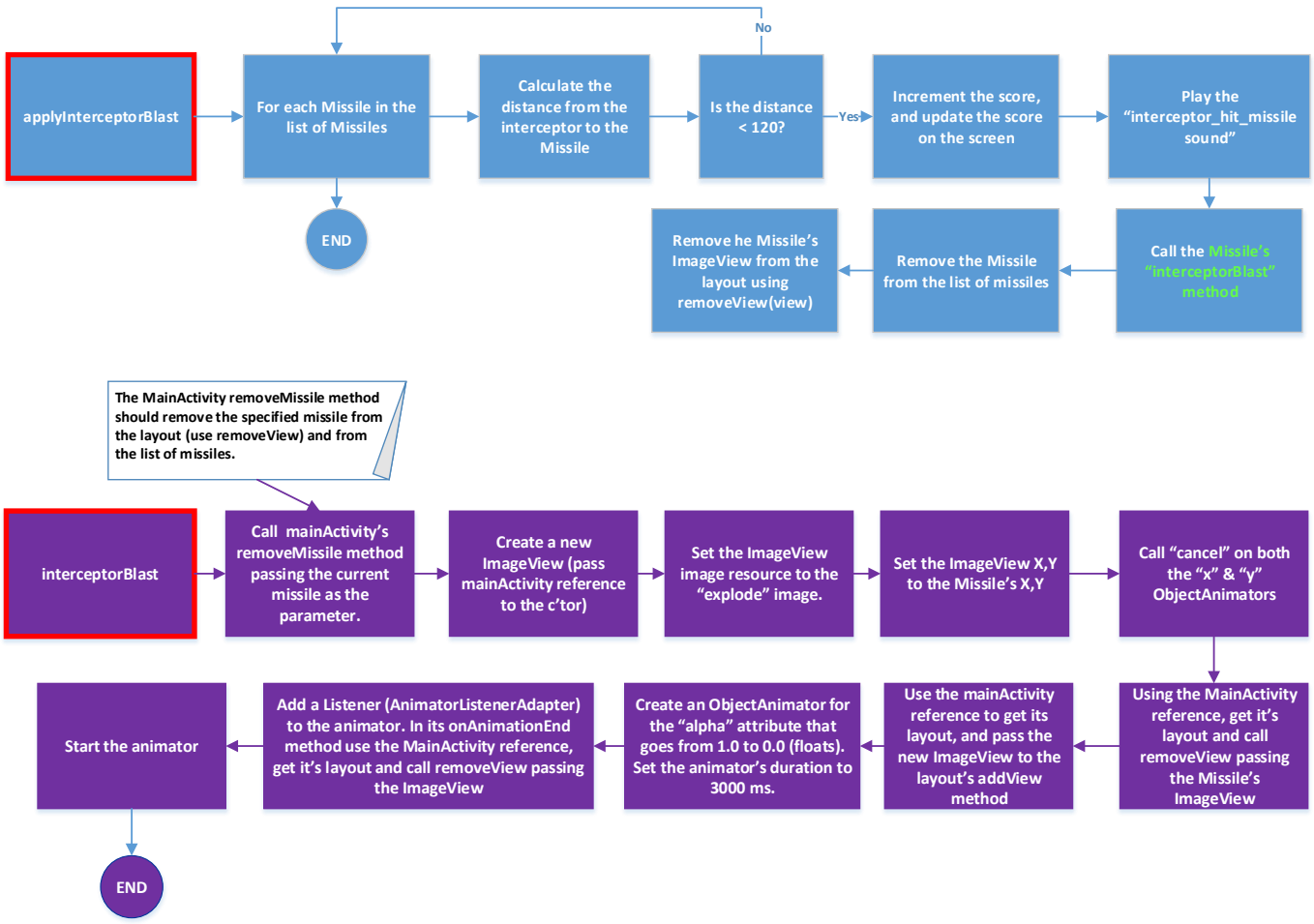

getSleepTIme
Generate a random number
if < 0.1 return 1 ms
else if < 0.2
return (0.5 * delayBetweenMissiles) ms
else return delayBetweenMissiles
END
launchInterceptor
Create a new Interceptor object passing it a MainActivity reference, the selected Base, and the touch X & Y
Play the "launch_interceptor" sound
Start the Interceptor's X & Y ObjectAnimators
END
Missile C'tor
Use the provided MainActivity and missile duration to set Missile's data fields
Create a new ImageView (pass mainActivity reference to the c'tor)
Set the ImageView image resource to the "missile" image.
Calculate the missile's start X value as (random number * screenWidth) and end X also as (random number * screenWidth)
Set the start Y as -100 Set the end Y as the screenHeight
Subtract half the missile ImageView's intrinsic width (imageview.getDrawable().getIntrinsicWidth()) from the start X and start Y
Calculate the rotation angle of the missile based on the start and end X,Y values
Set the ImageView's X to the startX, set the Y to the startY, set the Z to -10, and set the rotation to the angle you previously calculated
Use the mainActivity reference to get its layout, and pass the missile to the layout's addView method
Create ObjectAnimators - one for the "x" and one for the "y" values.
The begin/end values should be the start/end X & Y values.
The duration should be the current missile duration.
Do not yet start the animators
Add an UpdateListener to the "x" ObjectAnimator
END
onAnimationUpdate
Is the Y value is > (screenHeight * 0.85) ?
No
Yes
END
Call "cancel" on both the "x" & "y" ObjectAnimators
Call makeGroundBlast
Call mainActivity's removeMissile method passing the current missile as the parameter.
END
The MainActivity removeMissile method should remove the specified missile from the layout (use removeView) and from the list of missiles.
double angle = Math.toDegrees(Math.atan2(x2 - x1, y2 - y1));
// Keep angle between 0 and 360
angle = angle + Math.ceil(-angle / 360) * 360;
Float rAngle = (float) (190.0f - angle);
Interceptor C'tor
Use the provided MainActivity and endX & end Y values to set Missile's data fields
Create a new ImageView (pass mainActivity reference to the c'tor)
Set the ImageView image resource to the "interceptor" image.
Set the ImageView's startX to the X value passed to the C'tor. Set startX to (screenHeight – 70)
Subtract half the interceptor's ImageView's intrinsic width (imageview.getDrawable().getIntrinsicWidth()) from the end X and end Y
Calculate the rotation angle of the missile based on the start and end X,Y values
Set the ImageView's X to the startX, set the Y to the startY, set the Z to -10, and set the rotation to the angle you previously calculated
Use the mainActivity reference to get its layout, and pass the missile to the layout's addView method
Calculate the distance from the start x/y to the end x/y
Create ObjectAnimators - one for the "x" and one for the "y" values.
The begin/end values should be the start/end X & Y values.
The duration should be (the distance you previously calculated * 2)
Do not yet start the animators
Add an Listener (new AnimatorListenerAdapter) to the "x" ObjectAnimator
END
onAnimationEnd
Using the MainActivity reference, get it's layout and call removeView passing the ImageView
Call makeBlast
END


makeGroundBlast
Create a new ImageView (pass the mainActivity reference)
Set the ImageView image resource to the "explode" image.
Set the ImageView X,Y to the missile's X,Y
Subtract half the ImageView's intrinsic width (imageview.getDrawable().getIntrinsicWidth()) from the ImageView's X & Y
Set the ImageView's Z coordinate to -15
Use the mainActivity reference to get its layout, and pass the imageview to the layout's addView method
Create an ObjectAnimator for the "alpha" attribute that goes from 1.0 to 0.0 (floats). Set the animator's duration to 3000 ms.
Add a Listener (AnimatorListenerAdapter) to the animator. In its onAnimationEnd method use the MainActivity reference, get it's layout and call removeView passing the ImageView
Start the animator
Call mainActivity's applyMissileBlast method, passing the X & Y of the ImageView
END


makeBlast
Play the interceptor_blast sound
Create a new ImageView (pass the mainActivity reference)
Set the ImageView image resource to the "i_explode" image.
Set the ImageView X,Y to the interceptor's X,Y
Subtract half the ImageView's intrinsic width (imageview.getDrawable().getIntrinsicWidth()) from the ImageView's X & Y
Set the ImageView's Z coordinate to -15
Use the mainActivity reference to get its layout, and pass the imageview to the layout's addView method
Create an ObjectAnimator for the "alpha" attribute that goes from 1.0 to 0.0 (floats). Set the animator's duration to 3000 ms.
Add a Listener (AnimatorListenerAdapter) to the animator. In its onAnimationEnd method use the MainActivity reference, get it's layout and call removeView passing the ImageView
Start the animator
Call mainActivity's applyinterceptorBlast method, passing the X & Y of the ImageView
END


applyMissileBlast
Find the closest Base to the blast X,Y
Is the distance from the Base to the blast < 250?
Yes
No
Remove the Base from the list of Bases
Call the Base "destruct" method
If the list of bases is now empty, the game is over – call endGame
END


Base's destruct
Play the "base_blast" sound
Using the MainActivity reference, get it's layout and call removeView passing the Base's ImageView
Create a new ImageView (pass mainActivity reference to the c'tor)
Set the ImageView image resource to the "blast" image.
Set the ImageView's X,Y to the Base's X,Y
Use the mainActivity reference to get its layout, and pass the missile to the layout's addView method
Create an ObjectAnimator for the "alpha" attribute that goes from 1.0 to 0.0 (floats). Set the animator's duration to 3000 ms.
Add a Listener (AnimatorListenerAdapter) to the animator. In its onAnimationEnd method use the MainActivity reference, get it's layout and call removeView passing the ImageView
Start the animator
END



5) Remove Database Details

Remote Database URL: `jdbc:mysql://christopherhield.com:3306/chri5558_missile_defense`
Remote Database Username: `chri5558_student`
Remote Database Password: `ABC.123`
Remote Database Table: `AppScores`

Remote Database Table Columns:

1) DateTime — BigInt (Long) Not NULL ← Just use System.currentTimeMillis()
2) Initials — VarChar(3) Not NULL
3) Score — Int Not NULL
4) Level — Int Not NULL

Example Query for top 10 scores:

```
SELECT * FROM AppScores ORDER BY Score DESC LIMIT 10
```

Example Insert:

```
INSERT INTO AppScores VALUES (1582770779811, 'SRH', 11, 2)
```

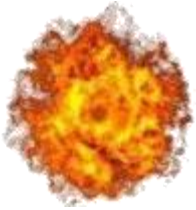
6) Provided Assets

Images (drawable):

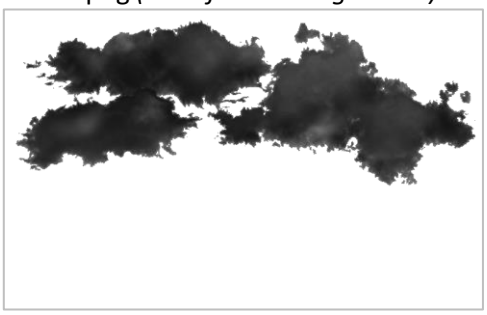
background.jpg *(background for all activities)*

base.png *(image for your 3 Bases)*

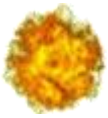
blast.png *(used when Base is destroyed)*



clouds.png *(used for scrolling clouds)*



explode.png (used for missile explosion due to interceptor or ground strike)



game_over.png *(shown when all bases destroyed)*

i_explode.png *(used when an interceptor explodes)*

interceptor.png *(image for interceptors)*

launcher.png *(used as app launcher icon)*

missile.png *(image for missiles)*

title.png *(used as game title)*

Sounds (raw):

background.mp3 ← Played throughout the game, started in SplashActivity

base_blast.wav ← Played when a base is destroyed

interceptor_blast.wav ← Played when an interceptor explodes

launch_interceptor.wav ← Played when an interceptor is launched

launch_missile.wav ← Played when a missile is launched from the top of the screen

7) Extra Credit Opportunities

There are 4 extra credit opportunities you can attempt which will enhance the quality of the game. These are optional and will not affect your grade should you choose to not do them. You can do any number of these 1, 2, all, etc. (however, numbers 2 and 4 are mutually exclusive).
Extra Credit is only awarded if the feature works as specified. You *must* indicate with your submission which if any of these extra credit opportunities you have implemented for them to be considered. None of these will be considered if you do not indicate which you have done.

1) **25 EC points**. Limit the number of interceptors allowed in flight to 3 at any one time. Attempts to add more than 3 live interceptors should do nothing. 1, 2 or 3 interceptors can be in flight at any time. When an interceptor explodes, it is no longer considered in-flight.

2) **10 EC points**. Enforce a minimum interceptor launch height. An interceptor cannot be launched if the touch point is lower than 80% of the screen height. Touching the screen in the lower 20% will do nothing – no interceptor launch.

3) **20 EC points**. Make the scrolling clouds fade in and out over time by slowly varying their "alpha" value. The scrolling clouds alpha value should slowly change from the lowest of 0.25 (nearly transparent) to the highest of 0.9 (nearly opaque) and then back again from 0.9 down to 0.25 – repeatedly. This should continue for the duration of the game.

4) **20 EC points**. Interceptors that explode near a Base can destroy the Base. As the game is defined, interceptor detonations do not consider the bases, only missiles. Here, when an interceptor detonates, you need to check if it was close to any missiles or any Bases. If a Base is within the blast radius of the interceptor, the base will be destroyed.

8) BONUS: Submission Date Extra Credit

This assignment is due in 3 weeks, same day as our class lecture (Thursday), by 11:59pm (as we have no Final Exam, you can work on this up to the date we *would* have had our final exam. **However** – if you submit in 2 weeks (Thursday), by 11:59pm you will get 10% extra credit (40 pts).

NO LATE SUBMISSIONS ARE ALLOWED FOR THIS ASSIGNMENT

Assignment Assistance

If you are stuck on an assignment problem that you have exhaustively researched and/or debugged yourself, you can email link to your project (in a ZIP file) so that I can examine the problem. NOTE that Android Studio projects cannot be emailed. You will have to copy the project to a file chare (Google drive, DropBox, etc.) and then send me that link.

All emailed assistance requests must include a detailed description of the problem, and the details of what steps you have already taken in trying to determine the source of the problem.

Note: To make your submission zip file smaller, before zipping your project file you must remove the ".gradle" folder (found in your project's root directory), and remove the "build" folder (found in the "app" folder in your project's root directory).

Submissions & Grading

1) Submissions must consist of your zipped project folder. For submission - before zipping your project file you must remove the ".gradle" folder (found in your project's root directory), and remove the "build" folder (found in the "app" folder in your project's root directory). Submissions not following these requirements will be penalized.
2) Submissions should reflect the concepts and practices we cover in class, and the requirements specified in this document.
3) **NO LATE SUBMISSIONS ARE ALLOWED FOR THIS ASSIGNMENT**
4) Grading will be based upon the presence and proper functionality of all features and behaviors described in this document.

NOTE

This assignment is worth 400 points. This means (for example) that if you get 89% on this assignment, your recorded score will be:

(89% * 400 points = 356 points)

If you do not understand anything in this handout, please ask.

Otherwise the assumption is that you understand the content.

Unsure? Ask!