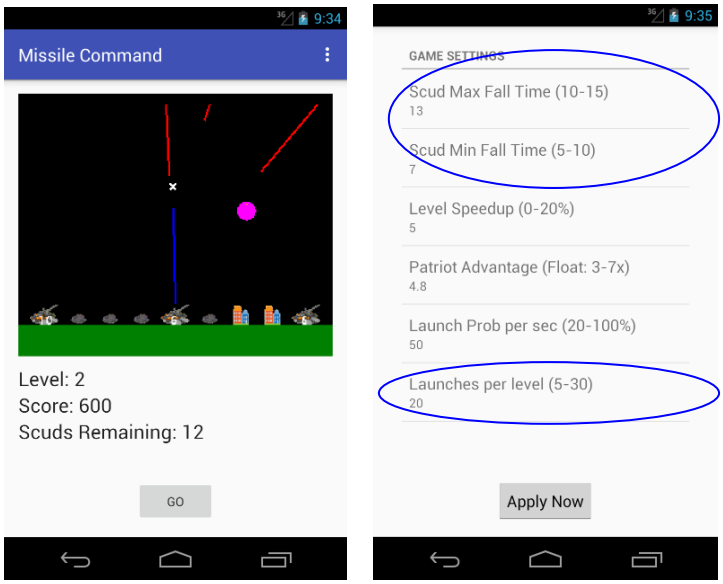
CSCD 372, Android Programming
Project: Missile Command

Missile Command is a 1980s arcade game by Atari that was also licensed to Sega. The idea of the game is to shoot defensive missiles (Patriots) at incoming enemy missiles (Scuds) in order to protect 6 cities (and 3 Patriot launchers) from destruction. The game ends when all 6 cities are destroyed. You can play an online recreation of the original game here: http://my.ign.com/atari/missile-command. Your task is to create your own Android version of the game. It should look something like this:



I will post my solution so you can try it out. You are NOT allowed to use an SDK other than Android for this project (e.g., Corona, LibGDX). I'm aware that there are Java sources out there for Missile Command. Please follow the rules specified in "Cheating FAQ" if you make use of such sources, and if you are in doubt about your use, just ask. If you want to make use of my bitmaps or sound files, just ask.

Following are specifications for the project, along with some hints on how to proceed. Read the specifications carefully, as each requirement has points associated with it. I am also providing the scoring sheet that I will use to grade your project. If you are uncertain about any of the requirements, please ask. I am distributing the project definition early so that you can make incremental progress as you acquire knowledge. I suggest that you get started as soon as possible in order to maximize your opportunities to receive help. Only bad things can come if you wait until the last week of the quarter to get started.

Specifications

1) The playing area must use a fixed aspect ratio and scale itself to the space available on the screen.
2) You can use bitmaps for cities and launchers, as I did, or simply draw them with primitives.
3) The game should initialize with 6 cities and at least 1 missile launcher. I used 3 launchers as in the original game. The launcher(s) should initialize with 30 shots total at the start of each level, distributed equitably. Note that the launcher(s) are resupplied with missiles at the start of each level. Each launcher should provide some kind of indication of the number of missiles left.
4) The game should launch a Preference-specified number of Scuds per round, and that should default to 20. My Scuds fall with a speed chosen randomly from a Preferences-specified range. You need not allow a range, but you must at least provide a Preference that controls the incoming speed on the first level. I also provide a Preference that controls how much the speed increases on each successive level, but you are not required to do that either.
5) You may fix the firing rate of Scuds or launch them according to some probability as I did. I have a Preference that allows adjustment of that probability, but you are not required to do that. If you fix the rate or the probability, a Scud should be launched, on average, about once every second or two. Scuds should originate from the top of the screen. My scuds originate from a random x location, but you may fix the origination point if you like.

6) Each incoming Scud should be directed towards a city or a launcher, such that they hit one of the targets if not intercepted. I pick one of the targets randomly, but you may cycle through them instead. You should target all launchers and cities, more or less equitably, so that the game can be lost. My Scuds do continue to target ruined cities.
7) When a city is hit, it is destroyed for the duration of the game. You must either erase it from the display or replace it with an indication of ruins. In my implementation a city can also be destroyed by a poorly aimed Patriot, but you are not required to do that.
8) If you provide more than 1 launcher: when a launcher is hit by a Scud, its missile count should drop to 0 for the duration of that level. If you provide only 1 launcher, reduce its missile count by 1/3rd, with truncation, when it is hit.
9) Patriots are launched by touching a destination on the screen. You must play some kind of a missile sound when Patriots are launched, but you need not have a sound for Scuds. You should drop a mark on the screen at the destination to provide some visual feedback as to whether the Patriot is going to intersect a Scud at the right time.
10) A launched Patriot can come from any launcher that has missiles left. I search for the launcher closest to the target destination, but you are not required to do that if you provide more than 1 launcher. The upward traveling line, however, should come from the launcher that will have its count decremented.
11) Patriots should move faster than Scuds, on the first few levels at least. I have a Preference that controls the speed advantage as a multiplier of the average Scud speed on the first level. You are not required to provide that. If you fix the speed advantage, please make them 5 times faster than Scuds. Patriot speed should NOT increase with the level.
12) When a Patriot reaches its destination, it should explode into a ring of some reasonable diameter. My Patriots explode in gradually increasing, and then decreasing diameters. You are not required to do that, but you are required to visually indicate the extent of the explosion. You should also play a reasonable sound when a Patriot explodes.
13) If a Scud is within the diameter of a Patriot explosion, it is destroyed. My Scuds also explode on destruction, or when reaching their target, but you are not required to do that. When one of my Scuds explodes it can take out a nearby Scud with it, but you are not required to do that.
14) You should provide some way to manually pause and continue the game. Note that the button I use for this changes its label to indicate the action of the button. The game should also pause between levels, with some kind of audio notification and requiring an un-pause action from the user to start the next level.
15) A configuration change (e.g., the home key) should also pause the game. It should pick up where it left off upon return (meaning you need to save and restore state), but should require some action from the user to continue.
16) There should be some kind of status display indicating the current level, the current score, and the number of Scuds remaining to be launched in the current level. When the last city is destroyed, there should be some indication that the game is over, and some kind of sound should be played. The player should be able to start a new game at that point.
17) You are allowed to restrict your solution to one orientation. If you allow screen rotations, make sure the game is playable in both modes.
18) Scoring should be as follows: 20 points for each Scud killed, 40 points for each Patriot left over at the end of a level, 100 points for each City remaining at the end of a level. My solution shows each of these score components being tallied separately at the end of a level, but you do not need to provide that kind of feedback. You might also notice that my implementation supports orientation changes but disables them while the score is being tallied. Feel free to ask if you want to see how I did it.
19) Your game must include a menu with an About selection that divulges your name, the year, and quarter. It must also contain a Settings selection that launches a preferences screen.
20) Your Preferences should include the two mentioned above (and circled): (a) Scud speed on the initial level. I specify this as the time it would take (in secs) to traverse the screen if falling straight down. You may specify that some other way, but it ought to be reasonably intuitive for the user. I allow a range of speeds, but a single value is sufficient. (b) The number of Scuds launched per level. This must be range controlled between 5 and 30. It is possible to range control an EditText preference, but I will also provide a couple of custom preference widgets that make it easy to control the user's input range.

Advice

- I recommend that you implement the playing area with a custom view, using either of the two animation methods discussed in class. If you use the first method (based on View), don't try to save the Canvas that is passed to onDraw() in a member variable for reuse at your leisure. Simply call invalidate() when you want to force a redraw.
- I have classes for City, Launcher, Scud, and Patriot that are maintained in a collection.
- If you use a simple timer to drive animation, a 70 msec or so update rate ought to be plenty fast enough.

Project Score Sheet

NAME:

Pts	Val	Description of requirement
	2	About box is present
	4	Game area is sized to display
	4	Game displays 6 cities and at least 1 launcher
	2	Either one orientation is disabled or both orientations are functional
	3	A mechanism is provided for the user to pause and resume
	4	Patriots are launched by touching a destination on the screen, rising in straight lines
	4	Patriot destination is indicated by touch
	4	Scuds are launched from the top of the screen, falling in straight lines
	3	Patriot count is refreshed to 30 at the start of each level
	4	Scuds target cities and launchers
	3	Scud hitting a launcher reduces its patriot count as specified
	4	Patriots explode when they reach their target
	4	Scuds are destroyed when caught in a patriot explosion circle
	4	Cities are destroyed when hit by a scud
	3	Destroyed cities are erased or replaced by ruins
	4	Game ends when all cities are destroyed
	3	Game pauses between levels
	4	Game state is saved and restored in paused state on configuration change (e.g. home)
	2	Game does not continue playing when backed out of
	6	Sound clips are played for patriot launch, patriot explosion, and game over (2 pts ea)
	3	Initial scud speed can be set by a preference
	3	Number of scuds per level can be set by a preference
	3	Preferences are range controlled, as specified
	3	Preferences show the current setting on the preferences screen
	3	Game level is displayed
	3	Number of incoming scuds remaining is displayed
	3	Game score is displayed
	3	Game score is computed as specified
	5	Overall playability, controllability, animation, paddle, bricks, and ball dimensions, etc.
	100	Total Score

Notes: