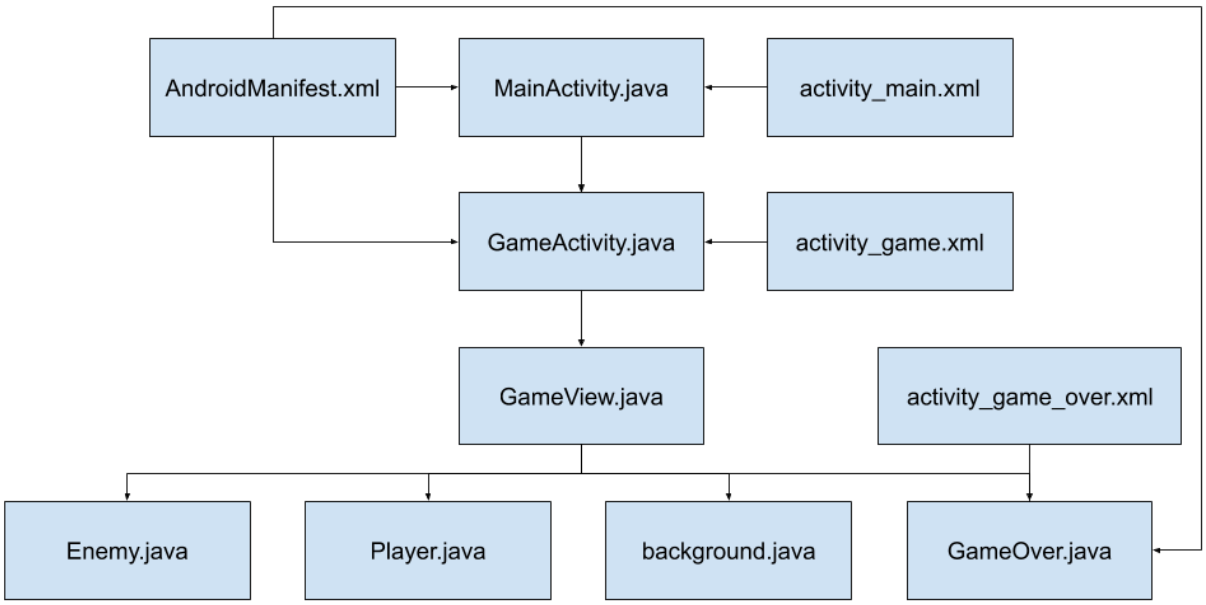
Knight Runner

Team Deep Blue
Sharith Gadamanna (sharithg), Charles Jamieson (cjam3), Ajay Ramnarine (ajram),
Dean Kiabi (mkiabi), Nikhil Gupta (nikhilgu)

GitHub: https://github.com/Jay-Rams/EC327_Final_Project

YouTube Demonstration: https://youtu.be/dJJ9N2u-VLo

Software Architecture:



Our Android app has an overarching file, the AndroidManifest.xml, that controls the activities and some of their characteristics. Upon bootup, the AndroidManifest.xml points to MainActivity.java which corresponds to the title screen. The activity_main.xml has many characteristics such as buttons and title screen graphics. MainActivity.java waits for the user to press the start button to move the game to GameActivity.java. GameActivity.java also contains an xml that contains graphics information. Here the phone screen size is taken and passed to the GameView.java which is the main part of our game. Here there is an instantiation of Player.java and instantiations of Enemy.java which contains information about the player and enemies such as there position and speed. Enemy.java also looks out for collisions with the player which upon decrease the lives of the player. Background.java is used in GameView.java to display a moving background giving our game a much more appealing visual. Lastly, if the user ends up losing all 3 lives, the game moves to GameOver.java which contains an xml with graphics and buttons to restart or return to the title screen.

Application Overview:

We created a 2D retro style side scroller game where your goal is to avoid oncoming enemies. With games like Flappy Bird being highly successful, creating a game of this nature seemed very fitting.

The application is useful for gaming on the go as the game can end quickly and can also be played repeatedly. It is also helpful for when one needs to pass time as the game has an easy restart function to start the game over whenever one gets a game over. The application is also child friendly, being that the purpose of the game is to avoid the oncoming enemies instead of attacking everything in sight.

Our audience outreach includes all ages. As stated above, the application is appealing to kids for its non-violence factor. For teens and young adults, the application will be appealing for passing time. For older generations, a game with retro styled graphics would appeal to their nostalgia. With everyone having access to smartphones, and with mobile gaming on a current rise in popularity, our application would be popular with people of all ages.

Our application is different from others graphically, utilizing sprites for all of the graphics; but is very similar to other side scroller games functionally. With the sprites being custom made to a more retro theme, fans of this type of graphics will be attracted to our application. Side scrollers are also a simple type of game where the user won't have to put much thought into playing, and with an easy to use interface to restart the game, the user will be more likely to play the game for longer periods of time.

Component Descriptions:

Processing:

The backend part of the code was created to work within the GameActivity.java activity. To make all parts of the game activity work, four classes are used: GameView.java, Player.java, Enemy.java, and background.java.

GameView.java connects all of the other classes to GameActivity.java. Within the class, the canvas that is drawn is implemented, as well as creating new instances of the player class, the enemy class, and the background class. Variables for the amount of hits taken, the amount of lives left, and the boolean to check on if the game is over or not are all created. This class also contains an update method to update the background, player, and enemies, and will also check if the player has been hit 3 times, resulting in the isGameOver boolean to be switched to true if they have been hit 3 times. An onTouchEvent method also exists in this class to check if the player is touching the screen or not, setting the player to move up the screen or down the screen respectively.

Both the Player and Enemy classes utilize the "Rect" class, which is used for collision detection by checking if the boundaries of a Player and Enemy are the same. If they collide, then the draw method in GameView.java is set to update the amount of lives left, and colliding 3 times results in the activity switching to GameOver.java. Both the Player and Enemy classes have an update method, which updates their positions based on the players speed. GameView.java will update the enemies speed as well as the movement of the background in the update method. The draw method within GameView.java uses the canvas to draw the enemies on the screen, display the current lives and time in seconds, and, if out of lives, changes activity to GameOver.java. Finally, in order to move the character, there is an onTouchEvent method, which uses a MotionEvent to check if the user is touching the screen.If they are, then the character moves up in the y-direction, otherwise, they move downwards. The background.java class uses a canvas to display the background image to fill the screen, and also makes the background loop as it moves.

GUI:

The interface was made to be simple enough for anyone to understand. It is comprised of three activities, each with their own xml files containing backgrounds and image buttons, and a day and night mode implemented into the overall styles.xml file.

Starting with the title screen, the user is met with a simple background and three buttons. The largest button sitting in the middle of the screen is labeled with the word “Start!,” which does exactly what it says. Once the button is pressed, the title screen activity switches to the actual game. The other two image buttons underneath the start button are labeled with a sun and a moon. The game is set to automatically detect when it should switch between a day and a night mode. The image button with a picture of a sun will set the game to be in day mode, and that will remain through the rest of the game once pressed. The image button with a picture of a moon will set the game to be in night mode, which will set all of the assets to a night mode version of the game.

Once the start button is pressed from the title screen, the game switches to the main game screen. There is not too much happening with the main game in terms of the graphical user interface. The background of the main game is set to move within a class named GameView.java, and will also change between day and night mode. A timer as well as the amount of lives the player has remaining will appear in the top right and top left of the screen respectively. The player and the enemies are made using custom sprites. The timer will increase in seconds as time passes, and the amount of lives will decrement as the player runs into enemies. Once the player has run into 3 enemies and they have run out of lives, the game will end and it will transition to the game over screen.

The game over screen will display a graphic at the top of the screen, containing the words “Game Over” with a sword going through it. The background, as with all the other backgrounds in the game, is set to change depending on if the user chose between day or night mode. There are two image buttons present in the middle of the screen. The image button with the label “Title Screen” will return the user to the title screen, where they can again choose to start the game or to switch between day and night mode. The image button with the label “Restart” will send the user back to the game screen with all 3 of their lives and the timer reset so that they can play the game again.

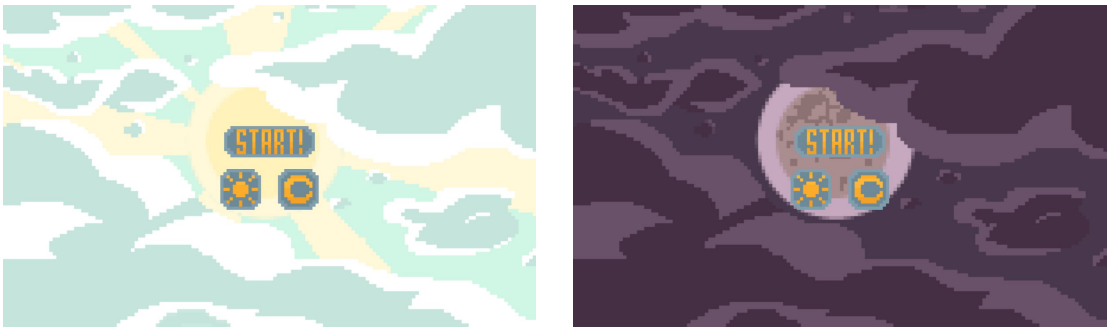


Figure 1: Screenshots of the day and night mode of the title screen, showing the three image buttons with one labeled “Start!,” and the other two labeled with a sun and a moon to represent day mode and night mode respectively.

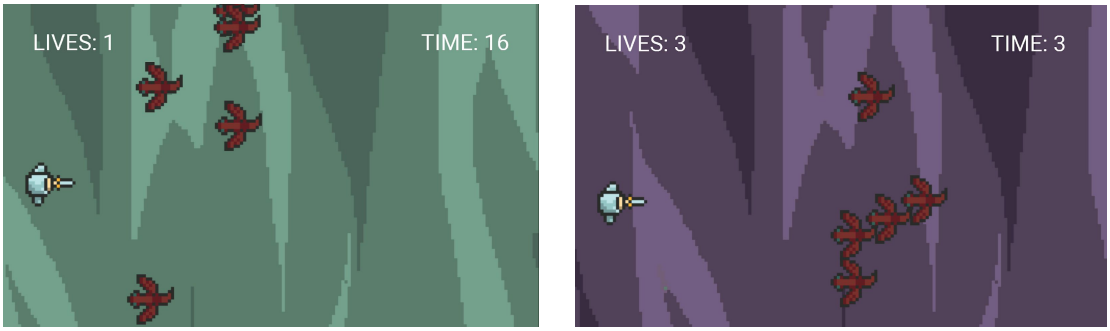


Figure 2: Screenshots of the day and night mode of the game screen, showing the amount of lives in the top left and the amount of time that has passed in the top right. The enemies and the player are also shown moving across the screen.

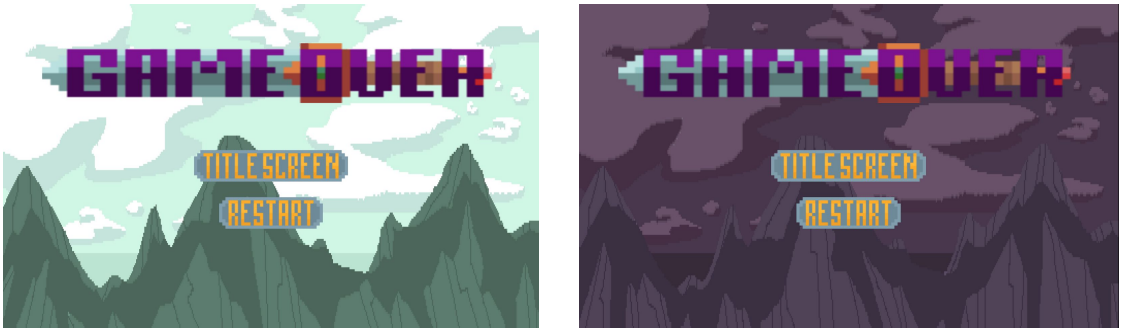


Figure 3: Screenshots of the day and night mode of the game over screen, showing the game over graphic at the top of the screen, as well as the title screen image button and the restart button right underneath the graphic.

Task Distribution:

Sharith Godamanna: Project Processing - Wrote the back-end code for the main component of the application. Created the different classes for the player and the enemy, as well as the game view class, which implements everything for the game activity. Came up with and implemented the idea for keeping track of the lives remaining that the player has, as well as implementing the timer on the screen so the player knows how long they have lasted without losing.

Charles Jamieson: Project Lead - Checked in with everyone to make sure that our project was meeting the criteria and to ask if they needed help with certain aspects of their jobs. Helped debug and problem solve the back-end code and got the scrolling background to work in the game. Researched many components of the project. Got the group synched together by troubleshooting problems with GitHub and made sure everyone could run the game on their computer by cloning the GitHub repository. Created the Youtube video and software architecture chart to demonstrate the project.

Ajay Ramnarine: Graphical User Interface - Created the sprites used for the buttons, the backgrounds, the player, the enemy, and other graphics in day and night modes. Wrote the xml files to create the GUI and wrote the code to connect the activities with each other. Helped debug some of the back-end code to work with the GUI. Did research on styles and created a night mode style for the game to run in a day mode and a night mode.

Dean Kiabi: Documentation - Managed all documentation and the marketing for the project. Assembled and documented all the back-end and front-end codes in gitHub. Brought the team together and discussed the effectiveness of the task distribution for each member of team. Ensured that each member had an equal right to exchange his ideas through the process of the project. Tried to create a friendly atmosphere in the project meeting periods.

Nikhil Gupta: Researched the android code for many parts of the components of the application. Helped the group understand the code he found during his research. Helped other group members when they were looking for help. Helped with debugging some of the back-end code as well as helping to implement the timer in the main game.

Assessment:

Sharith Godamanna (19%), Charles Jamieson (19%), Ajay Ramnarine (28%), Dean Kiabi (17%), Nikhil Gupta (17%)

Project Timeline:

The team agreed to meet every week on Saturday or Sunday for 2 hours to exchange ideas regarding different components of the project and the implementation of those ideas.

- On March 30th,
  The team met to brainstorm different projects and the possible responsibility and task distribution that each member should take.
- On April 6th,
  Finalizing the project selection and task distribution.
- On April 13th,
  Checking the process and the pace of the project, and ensuring that each member was committed to the task.
- On April 20th,
  Put together all the documentation including the back-end and front-end for the game, and discussing the marketability of the game. Created GitHub accounts to share all the components of the code. Packaged all the code into the android studio.
- On April 27th,
  Testing the code, debugging process, fixing the errors, and making sure that the game doesn't use any unnecessary memory space. Adding more attractive custom graphic designs to make the game appealing, easy to use, and understand. Implemented a lives counter and a time survived counter to the background of the game.
- On April 30th,
  The team met for the last time for the finalization of the project, checking if the game functionality was persistent and all the code was well-documented. Discussed the possibilities to improve the functionality and marketability of the game.