Table of Contents

Initial Program Specs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

Work in Progress Report. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2 - 4

Scratches . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

Disclaimers/Bugs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

Note to Future Programmers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .5

Journal . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7 - 9

Sources. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

Lessons Learned. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

“User Manual” is within folder LOKO/FINAL PACKAGE. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

“PolyGone” cloned folder is within folder LOKO/FINAL PACKAGE. . . . . . . . . . . . . . . . . . . . . .

Initial Program Specs

NAME:Daphne Lai and Mila Jokovic

PROGRAM OVERVIEW:

An intuitive reflex based game that encourages the players’ concentration and simple counting skills. By recognizing shapes and the amount of sides it has, the player needs to be able to react to the amount of sides and reach a higher score under a certain amount of time.

DESCRIPTION OF USER INPUT:

Keyboard input is the main input needed from the user.

Enter key & Number Pad - during the game, the user needs to press the number that is the same amount of times as there are sides of the polygon, once they do that, they need to press enter in order to submit their answer.

Mouse input is needed for the user to be able to navigate through the game.

DESCRIPTION OF PROGRAM OUTPUT:

The program will begin with a menu screen so the user can change the volume settings, go to the game, find a game tutorial, or see credits. When the player begins a game, a timer will appear at the top counting down from one minute.

While time is of the essence, the player will see a type of polygon appear at the centre of the screen. After they’ve submit the number of sides and “enter”, the current shape will fall and a new one will appear with different amount of sides.

If they enter the wrong number, time will be lost and a red “X” will show that they are incorrect. Otherwise, a green checkmark will show.

When the timer is up there’ll be an end screen while displaying your current score asking the user to either retry or return to menu.

Work In Progress Report

**Loko**

<https://github.com/DaphneLai/POLYGONE-Final.git>

**Major developments/breakthroughs(reference specific code please):**

**-Randomization of shapes:** When the user presses the “down” arrow button, a random shape and colour of shape appear on the screen. (gdx.screens > scrGame.java > lines 161 - 179)

-**Game menu:** The game’s home screen/menu is finished, and the buttons all work. Since we have not added music yet, the sound buttons don’t perform any tasks, but recognize when they are pressed. The tutorial button leads to a tutorial. (gdx.screens > scrMain.java)

**-Manually switching between screens:** We can switch between game screens before running. For example, we have assigned numbers to each screen.When you input the number of screen you want (line 39, “update state”) and then run the desktop launcher, it will display the screen that you have chosen. (gdx.Menu > GamMenu.java > lines 22 - 59)

-**Created graphics:** We have made different colours and types of shapes for our game. We have also created images for buttons, titles, and have chosen background images as well.

-**Reading keyboard input and displaying it on the console:** We can input numbers from the keypad and they will be outputted into the console. (gdx.scratch > NumInput.java > lines 22-64)

-**Making shapes move up and down:** We can move the shapes down, off screen, and then back up. (gdx.objects > Shape.java )

**-Started a countdown timer:** We have borrowed code from Ameer for our countdown timer. We now have a timer that will close the screen as soon as the time is up. However, we still need to add the numbers counting down on the screen, and be able to program the shapes to stop appearing once the timer has finished.

**Major Challenges/setbacks( reference specific code please):**

**-GitHub:** We couldn’t figure out how to push/pull with GitHub, making it difficult to work on our project together.

-**Scratches:** We didn’t quite understand the concept of scratches and how to incorporate them into our code. We decided on making a package called “scratches” and we would do our “test runs” on there. However, we couldn’t run our scratches, and we didn’t know why, so it was difficult to test what we had to see whether or not it worked. (gdx.scratches).

-**Centering without hard coding:** The shapes were different sizes and so they would appear in slightly different positions. We remade the shapes to fix this. There is no specific code for this, we simply recreated our shapes on Adobe Illustrator and made sure to have the same dimensions.

-**Creating buttons:** We had to create and adjust button class, as well as research and understand many different concepts such as changing and modifying the class. (gdx.objects > Button.java)

-**Cleaning up code:** We tried to make code as efficient as possible. This proved to be difficult at times because some code couldn’t really be fixed up, or else it would interfere with what we have done.

-**Problems with orthographic camera:** The orthographic camera is confusing and is conflicting with textures. We don’t know where the origin should be. (gdx.screens > scrGame, scrTutorial)

-**Questionable shapes:** After a fair amount of clicking the “down” arrow button, the shapes don’t come back up anymore.

**Any modifications to your specifications/release schedule:**

The release schedule has been modified quite a bit, but most of the revisions are simply changes in the order of the releases, that made more sense to us. Our first release schedule had its last release as 3.0, but our updated release schedule has been modified to have its last release as 3.3.

|  |  |  |
| --- | --- | --- |
| **Release Name** | **Old Release Schedule** | **Modifications** |
| 1.0 | Randomizing shape appearances | Upload and show textures |
| 1.1 | Orthographic camera | None |
| 1.2 | Make shapes move up and down and change | Background image for game screen |
| 1.3 | Counting keyboard input | None |
| 1.4 | Timer | Make shapes change |
| 1.5 | Lose seconds when user plays out of bounds | Make shapes move up and down |
| 1.6 | Point counter | Timer |
| 1.7 | **End screen** | Loser seconds when player is out of bounds |
| 1.8 | Point display + amount of seconds lost | Put in simple graphics |
| 1.9 | Retry button | Point counter |
| 2.0 | Home button | **Home screen** |
| 2.1 | **Home screen** | Game title |
| 2.2 | Game title | Play button |
| 2.3 | Play button | Sound and music button |
| 2.4 | Sound and music button | Tutorial button |
| 2.5 | Tutorial button | **End screen** |
| 2.6 | **Tutorial screen** | Point display + amount of seconds lost |
| 2.7 | Add music | Retry button |
| 2.8 | Enhance graphics | Home button |
| 2.9 | Add sounds | **Tutorial screen** |
| 3.0 | Record last score | Add music |
| 3.1 |  | Enhance graphics |
| 3.2 |  | Add sounds |
| 3.3 |  | Record last score |

Description of your scratch/test program:

**Describe the generic concept you needed to test out:**

-We needed to test out the randomization and movement of shapes, since the concept of the game is to input the amount of sides the current shape on screen has. We also needed to test out whether or not the program could read the user input, so that once the user inputs the amount of sides the displayed shape has (correctly), the program will be able to move on to a new shape and keep the game going.

**Describe the code and the lesson that you learned from it:**

-Keyboard input code: We learned that there are prefabricated methods for keyboard input and we learned the keycodes for each number and the enter button. We’ve also learned that you cannot nest the “if” structures when checking for whether or not someone has pressed the “enter” button. If this is done, the program is checking for whether or not someone has pressed a number, and the enter button at the same time, which isn’t effective or possible.

-Button classes: We learned how to make classes in general, and this code really expanded our research skills, since it required a lot of searching around different websites.

**Describe any challenges that you enjoyed in integrating this scratch code into your major project:**

-Mila: I had some trouble understanding the concept of libgdx and it’s basic factors. It was difficult to wrap my head around them. Some challenges that I had were understanding how to make the code we learnt in class work in libGDX. A scratch I had difficulty with was the keyboard input, because at the beginning I didn’t know that there were some methods already created for it, and that you needed “keycodes.” I enjoyed these challenges because it made me work harder to understand and it taught me not to be afraid to ask questions.

-Daphne: I enjoyed getting programs to work. Implementing code made me have to think abstractly and be able to adapt to strange concepts. I also had to research quite a bit, which challenged me in a good way, since I had to make use of all of my resources effectively.

**Peer Assessment:**

Daphne : 100

Mila: 100

Scratches

**Scratch #1: Num Input:** This scratch tests out the ability to read in a user’s keyboard input. The code reads in the input as a string, so that when a user is inputting double digits, it keeps adding onto it. Once the user is done inputting their numbers, they have to press the “enter” button, to be able to submit their answer. This scratch also displays the input on screen as the user is typing. This makes the graphics of the game more fun and interesting. Once they press enter, the numbers on screen disappear, ready for the user’s next answer.

**Scratch #2: Countdown:** This scratch is our timer. The timer can manually be set to any number you want. This one in particular counts down from 60 seconds. What this code does is take away one second while a boolean (isOn) is true. Once the timer is up, the screen will close on its own. This code also outprints the timer on the screen, in a font we’ve chosen. We borrowed this code from Ameer.

Disclaimer/ list of known bugs

PolyGone has gone through multiple test runs and has little errors for the first two rounds.

Errors we’ve experienced is that when one mutes the music, the music stutters instead of pausing completely (only has happened on Daphne’s computer).

Another bug is that if the player presses enter at the last second of the game, the shape will fall (as it’s supposed to). But when the player goes to replay another round, the shape from the previous game will continue to fall as it was interrupted when the player went to the end screen.

For the code itself, there will possibly an indicator that screens under the gdx.screens package are deleted. We have been dealing with this for the last few weeks since we did not know how to fix it. Luckily we managed to understand that somehow the files were deleted from the repository but are able to be updated and can still run. As you (Mr. Grondin) are not changing or pushing code, this won’t affect you. But we still count this as a problem.

Note to Future Programmers

Programming is not easy for everyone but if you have a passion for it, you have to study for it. I (Daphne) enjoy programming (especially when it works) but I am not that skilled it. Instead of treating it as a new concept in general I treat code as if I am learning a new language. The best way to learn is to use Google… All the time. For a better experience understand that code isn’t a test where you can’t ask for answers. The internet and other programmers can help you to grow and understand.

There were many times when I, (Mila) was very frustrated with myself, as well as the code. However, this didn’t mean that I gave up. It’s important to keep working towards your goal, and instead of being angry with yourself or your program, you should put that energy towards searching for answers. Also, it’s okay to ask questions. Sometimes you can’t figure everything out on your own and that's okay.

Journal

**December 10, 2017**

Our project has been introduced and we have chosen each other as partners. We went through many different game ideas, and are yet to find one we think is the best. We have also gone through all of the required parts of the project and have created all of the necessary documents.

**December 11, 2017**

We have agreed on what our game will be and have created a release schedule that we think is manageable. We have also started to make our graphics using Adobe Illustrator, and have agreed upon a colour scheme for our game.

**December 14, 2017**

We realized that in order for our game to be able to randomize shapes and colours of shapes, we had to use a 2D array. Grondin has suggested that we change the keyboard input as well. Instead of using the spacebar to tap how many sides a shape has, we should get the user to input to number of sides with the keypad and then press enter (to avoid over-tapping). We attempted to expand our knowledge on the 2D arrays, so that we could accomplish randomization. We had never used 2D arrays before, so we seeked help from the class website, and were able to fill the array with different shape variations.

**December 15, 2017**

We were working on randomizing shapes by creating two “for” loops. One populated one part of the 2D array with a shape, and the other populated the other part of the array with a colour. This task was a little difficult, and took some time before we were able to figure it out, but it worked in the end. However, we still need to figure out how to get the program to randomize the shapes by itself, one at a time.

**December 18, 2017**

We tried to understand GitHub and the process of pushing and pulling, but were unable to. We spent most of our time working on solutions to this problem but we didn’t find any. It was very tiresome, as we aren’t able to work on the game together, and instead have our own separate libgdx projects to work on. Once we realized that we couldn’t fix our issue ourselves, we put most of our focus on creating a menu and the randomization, until we could get help.

**December 20, 2017**

While working on shape randomization and the creation of a menu, we also tested pushing and pulling to and from the GitHub repository once more. However, we struggled in this and it failed to work several times. This was a very frustrating process. Even though we asked for help on many occasions, no one was able to solve our problem. The internet also provided little help. The problem with GitHub is still unclear, despite constant research.

**December 22, 2017**

While we were working on the game menu and the randomization of shapes, Mr.Grondin helped us with our GitHub problem. We were able to clone, push, and pull successfully. We also learned that since the school’s computers are slow, we should wait a while before pushing. If we don’t allow the programs to catch up, GitHub might push the files without updating some of the information, resulting in major conflicts.

**December 26, 2017**

We wanted to work on our code at home, but found it rather difficult. Mila couldn’t clone the game from the GitHub repository, and Daphne couldn’t run the file. We tried many times to fix this problem but found it rather confusing. We worked on trying to solve the problem for several days, so that we could start working on code once again.

**December 30, 2017**

We solved our problems are now able to work on the game from our laptops. As it turns out, a gradle is required to be able to clone a libgdx project, even on higher quality computers. Once a gradle was downloaded, it all worked smoothly. It was quite frustrating that we weren’t able to work on our code at all up until this point. The issues that were stopping us from progressing were some problems with Mila’s gradle and Daphne’s Netbeans was having conflicts with java.

**January 1, 2018**

The game menu has been finished, although we are still working on the buttons. We are also focusing on keyboard input. We have made a scratch and are trying to read in the keyboard input, and display what the user has inputted on the game screen, after they’ve pressed enter. It is proving to be quite difficult and frustrating, since there isn't much help to be found online sp far, but we will keep searching for solutions.

**January 2, 2018**

We have started working on the more important aspects of our game. We are working on changing the shapes manually on the screen. We have also figured out a bit of keyboard input, by using “(Gdx.input.isKeyJustPressed(Input.Keys.NUMPAD\_1).” We found this code on a youtube video about libgdx. After searching the internet and some trial and error, it was able to run. Now we have to work on outputting the input of double digits. However, we are confused about “scratches” and how they work. We have made a separate scratch package, but aren’t sure whether this is the right way to go about making scratches.

**January 3, 2018**

We can now change between shapes by pressing the arrow keys but have not yet randomized between colour or order. Now, we are working on getting the shapes to change by themselves. The keyboard input is still in progress. We think we might have figured out how to read input once the user has pressed the “enter” button (by using “if” statements), however, the program will not output the keyboard input like we want it to. For some reason, the scratch we are using for keyboard input, will not run. All that is displayed is a black screen, even if we have put in a background image. Therefore, we can’t test whether or not our code works. We are working towards solutions.

**January 4, 2018**

We have managed to get the shapes (not colours) to change randomly only by pressing the right arrow key.

**January 5, 2018**

We cannot check whether or not the code for the keyboard input works for sure, because the program will not run. To try and solve this problem, we’ve created a new project and are working on there. We think that part of the problem is because the scratch is not in the “screens” package. Although there’s a slim chance that that is indeed the issue, we don’t have any other ideas on what it could be. However, the screens on the new project will run for 5 seconds and then stop.

**January 8, 2018**

We’ve figured out how to run the scratches and can now test whether or not the keyboard input works for sure.

**January 9, 2018**

We can now handle keyboard input with the “enter” key. When someone enters a number and then presses “enter,” it will outprint the number into the console. We did this using multiple “if” structures. We are also able to make the shapes move down and off the screen. This will be needed for the transitions between different shapes. We received help from classmates to achieve these things. We are now moving onto creating a timer for our game, and getting the shapes to move up.

**January 10, 2018**

The shapes can appear on screen, move down and off screen, and then move back up again as a different shape. We achieved this by asking for help from other classmates (Joel). We also have started the timer with the help of Ameer’s code.

**January 11, 2018**

The countdown timer has given us some difficulty. The code is fine, but the scratch was not being recognized as a screen. We solved this problem by deleting unneeded code such as “extend ApplicationAdapter” and found out that we had two “render” methods, which were confusing the computer. We have also started a point counter, and are researching it.

**January 12, 2018**

We’ve printed our first and only WIP!

We have made the timer appear on screen and have integrated the tutorial screen. We are now working on displaying keyboard input on screen so that once the user inputs their number, they’ll have a side-by-side of their answer vs the correct answer. We are working on putting a “back” button into the tutorial screen.

**January 13, 2018**

We have made the end screen, and we have also implemented the timer into the game, so that once a minute is up, the screen will be redirected to the end screen. Before we could implement the timer, we needed to make sure the numbers were nice and clear to read. The numbers that showed up on screen were blurry and very pixelated, so we had to research how to avoid that problem. We noticed that the thicker fonts were naturally blurrier onscreen, so we went with a thin font that still matched our game. Now the timer is nicer to look at. We have also added music, and now our “mute” and “sound” buttons work!

**January 14, 2018**

We are working on displaying the keyboard input on screen. Yesterday, the numbers would be flickering on screen and would overlap one another each time a new number was inputted. Now, however, the numbers will only pop up for a split second and then disappear. We are not sure when or how this change occurred. We are now working on how to get the number to appear on screen until the user has pressed the “enter” button. Update: We have achieved this, and are now going to implement it into the game.

**January 15, 2018**

We have integrated the keyboard input, and now the shapes on screen change once someone has inputted their answer. We have started the score counter. So far, there haven’t been any complications or frustrations!

**January 16, 2018**

Our GitHub malfunctioned. For some reason, it didn’t “pick up” on the last time that we committed and pushed our updated code. When we pulled, it gave us an older version instead of the newest, so now we’re in a pickle. We are now trying to work on making our array declarations shorter by making a “for” loop to go through all of our shapes and shape variations (until we can get the newest version of our game back).

**January 17, 2018**

For some unknown reason, when we commit, the GitHub automatically checks off the “delete” box for certain files. Therefore, it didn’t commit our latest changes. We now have to commit our entire game first, then once we have done that, we need to commit separate files and then push. It’s a frustrating process. We are also trying to display the score on screen, and are fixing up our end screen.

**January 18, 2018**

We have finished the point display, and have also fixed up our method for checking whether the player is correct or not. We are now going to work on the finishing touches of the game, such as, displaying the final score on the end screen and adding sounds for when someone gets the answer correct/incorrect. Once that is finished, we will be done the basics of our game!

**January 19, 2018**

We have made our “points” into a button, so that it can “cleanly” be added everywhere we need it throughout our code. We have also displayed the score on the end screen and gotten it to reset every time the player presses the “retry” button. We have noticed that if the player inputs a number and presses enter at the very last second of the game, the game timer cuts off a few seconds. Now, we are working on cleaning our code up, and fixing minor errors, or things we would like to run smoother/differently.

**January 20, 2018**

We have added sounds into the game, so that when a player gets the answer right/wrong, a sound plays. We have also fixed the issue we had with the timer. We have also made our tutorial screen read the text off of a file instead of a picture. This, however, is weird because the text displayed on screen is not formatted the way it is on the actual file.We will work on this.

**January 22, 2018**

We’ve finished our game! Our code is cleaned up, and our game runs the way that we want it to.

Websites/Sources

**Hex-RGB to Opengl:**

<https://openglcolor.mpeters.me/>

**Colours for shapes:**

<http://paletton.com/#uid=54K0u0kbhVY00++5LZGg1PXkbGF>

<http://paletton.com/#uid=75o0u0kdyWb00007RZqiiR8mtJ3>

**2D Array:**

<https://stackoverflow.com/questions/12231453/syntax-for-creating-a-two-dimensional-array>

**Keyboard input and output on screen:**

<https://github.com/libgdx/libgdx/wiki/Event-handling>

<https://www.youtube.com/watch?v=IsYhkng3r1k> (Timestamp - 5:45)

**Timer and Scoring (Didn’t use these in the end, but still helpful)**

<https://www.youtube.com/watch?v=gqxkeKaw1MY>

<https://stackoverflow.com/questions/25809293/java-libgdx-creating-and-displaying-a-stopwatch> (timer code)

<https://www.youtube.com/watch?v=7idwNW5a8Qs> (3.15, creating label)

**Fonts + Displaying Files/Fonts**:

<https://stackoverflow.com/questions/33633395/how-set-libgdx-bitmap-font-size>

<https://www.youtube.com/watch?v=vZElAQUZLSg>

<http://www.angelcode.com/products/bmfont/> (bitmap font creator)

<https://stackoverflow.com/questions/28908629/android-java-libgdx-text-file>

**Score:**

<https://www.youtube.com/watch?v=gqxkeKaw1MY>

<https://www.youtube.com/watch?v=cGqq59-Kd7Y> - displaying score

**Sounds:**

<http://soundbible.com/free-sound-effects-1.html>

**Websites in general:**

[https://stackoverflow.com](https://stackoverflow.com/)/

[https://libgdx.badlogicgames.com/](https://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/g2d/BitmapFont.html)

**Youtube channels:**

**-Gamefromscratch**

**-Awesome tuts**

**-Brent Aureli’s - Code School**

Favourite Lesson Learned

The favourite lesson I’ve (Mila) learned, that we haven’t learned in class, was working with fonts and input/output. I found it interesting to research different ways of doing this and finding the way that worked for me. The way that I’ve chosen to do it, is using the BitmapFont. You have to declare a new BitmapFont, and then using the BMfont program I found online, select and download a font you want. Once you’ve done this, you put the file into your assets, and grab it from there using: Gdx.files.internal("NameOfFontFileHere"). I found this entertaining, and enjoyable even. Another favourite lesson I learned, was how to display the font on the screen, whilst the user was typing. While typing, their input would appear on screen, then disappear when “enter” was pressed. The way I achieved this was by resetting “nNum” to zero, each time the “enter” button was pressed. Then I put the line of code that was outputting the numbers, outside of all of the “if” structures that read our input, so that it’s always outprinting.

I learned a lot of things while trying to debug our code and optimizing out code. Something I learned was integrating string manipulation for our textures. I actually was very happy that after we were told a more efficient way to load our shapes I was able to figure out loops and strings to save ourselves over 100 lines of code! Not only that but it helped me realize that there are so many ways we can use basics skills we learned earlier in the semester to use in our code today. That’s why I love code, there’s always more than one way and there’s always a more efficient way, you just need to take the time to learn it.