|  |  |
| --- | --- |
| **Task** | **Point** |
| Structure Chart | 6 |
| Data Dictionary | 3 |
| P-spec | 5 |
| Presentation | 2 |
| **GROUP TOTAL** | **16.0** |
| Individual Paragraph for ***Steven, Kyle, Kyle, Jonah*** | 2 |
| **TOTAL *kwasniewskik (-2), missing time log, Grade: 16*** | **18** |

***You need to read the requirement on D2L for this deliverable carefully. For data dictionary, you need to include the following information:***

* + - ***Name of the data item***
    - ***Description of how the data item is used***
    - ***Data type***
    - ***Initial value***
    - ***Range of values***
    - ***The functions/modules using/referencing the data item***

**Structured Analysis – Reengineering Gorillas**

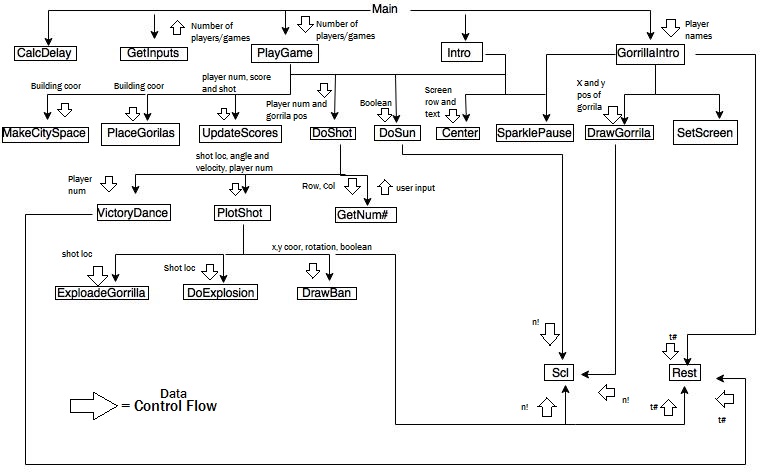
*Team Dank: Steven Hartnett, Kyle Hoffhein, Kyle Kwasniewski, Jonah Tollefson*

1. **Introduction**

Copyrighted by IBM Corporation in 1991; Gorillas is a simple game that pits the player up against the system or another player in a battle to defeat each other. The basic concept of Gorillas is that two players, play as gorillas and try to defeat each other by throwing exploding bananas at the other player. Gorillas follows the pattern of a physics shooting game; that is to say that the players throw their bananas at each other, but both must account for the wind speed, gravity, terrain, and the angle and initial power of their throw if they are to hit their target. Two important concepts behind the system to note, is the destruction of the terrain when hit with an exploding banana, and the AI system that adjust its difficulty depending on how well the player has been landing their hits. The game allows for two player, or play against the AI. Gorillas is indeed an older game, over two decades old, and the original source code is becoming outdated.

1. **Structured Analysis**

**2.1 Structure Chart**

****

**2.2 Data Dictionary**

GorillaX = (1,2)

GorillaY = (1,2)

LastBuilding = integer

Pi# = 3.14158 \*\*constant unless pi changes

LBan&(x) = Graphical picture of banana Left

RBan&(x) = Graphical picture of banana Right

UBan&(x) = Graphical picture of banana Upper

DBan&(x) = Graphical picture of banana Down

GorD&(120) = 'Graphical picture of Gorilla arms down

GorL&(120) = 'Gorilla left arm raised

GorR&(120) = 'Gorilla right arm raised

gravity# = integer =\*\*speed in meters per second of gravity for game

Wind = {-5,5} \*\* speed in meters

'Screen Mode Variables

ScrHeight = (350 | 200) in pixels

ScrWidth = (640 | 320)

Mode = {0-9} \*\* one integer

MaxCol = (80 | 40)

'Screen Color Variables

ExplosionColor = 2 ?

SunColor == ??

BackColor = (1 | 0)

SunHit = boolean

SunHt = (39 | 20)

GHeight ( 12 | 25 )

MachSpeed = speed that banana has to travel for it to not show on screen = calcdelay

|  |  |
| --- | --- |
| *Name* | Gorilla |
| *Is composed of* | GorD&(120) | GorL&(120) | GorR&(120) |
| *Range of values* | Can change 120 in game to change size of arms. |
| *Volume frequency* | Drawn at rate of variable calcdelay / second |
| *Source/destination* | “Main” method |

|  |  |
| --- | --- |
| *Name* | Banana |
| *Is composed of* | LBan&(x) | RBan&(x) |DBan&(x) | UBan&(x) |
| *Range of values* | X = {0-120} |
| *Volume frequency* | Drawn at rate of variable calcdelay / second |
| *Source/destination* | “Main” method |

|  |  |
| --- | --- |
| *Name* | Screen Mode Setup |
| *Is composed of* | ScrHeight = (350 | 200) in pixels  ScrWidth = (640 | 320)  Mode = {0-9} \*\* one integer  MaxCol = (80 | 40) |
| *Range of values* | Based on the screen orientation. |
| *Volume frequency* | N/A |
| *Source/destination* | “Main” method and “Intro” Subroutine |

|  |  |
| --- | --- |
| *Name* | Screen Color Variables |
| *Is composed of* | ExplosionColor = 2  SunColor == ()  BackColor = (1 | 0) |
| *Range of values* | Can change in program, for programmer’s fun |
| *Volume frequency* | Drawn at rate of variable calcdelay / second |
| *Source/destination* | “Main” method |

**2.3 P-Spec**

|  |  |  |  |
| --- | --- | --- | --- |
| **Modular name** | **module description** | **preconditions** | **post condition** |
| DoSun | Draws the sun at the top of the screen. | Banana is in mid-flight. | Banana is still in mid flight |
| SetScreen | Sets the appropriate color statements | Must be run during start up. | The color palette is set |
| EndGame | UNUSED | UNUSED | UNUSED |
| Center | Centers and prints a text string on a given row | The system has text to display | Text is centered on the screen. |
| Intro | Displays game introduction | Game parameters selected | Game begins |
| SparklePause | Creates flashing border for intro and game over screens | Must be in either the intro or game over screen | A flashing broader is drawn. |
| GetInputs | Gets user inputs at beginning of game | Game parameters have been filled in. | Game parameters are set. |
| PlayGame | Main game play routine | Game parameters have been set. | The game is over |
| DoExplosoin | Produces explosion when a shot is fired | A banana has hit something. | An explosion is drawn |
| MakeCityScape | Creates random skyline for game | A game has been started. | Buildings and wind are drawn to the screen. Gorillas are ready to be placed. |
| PlaceGorilla | Puts the Gorillas on top of the buildings. | Must have a cityscape to be placed on. | Gorillas are placed and drawn on the top of their respective buildings, gameplay begins. |
| GorillaIntro | Displays gorillas on screen for the first time allows the graphical data to be put into an array | Game parameters were set. | Game will begin. |
| Rest | pauses the program | The game is executing | The game is resting to accommodate computational speed. |
| VictoryDance | gorilla dances after he has eliminated his opponent | A gorilla was killed in action | A point is given to the surviving gorilla who performs a dance celebrating the demise of his opponent. |
| ClearGorillas | UNUSED | UNUSED | UNUSED |
| DrawBan | Draws the banana | Player or bot has input the parameter of their throw. | The banana fly’s across the screen. |
| Scl | Pass the number in to scaling for cga. | System is in startup | Screen is properly scaled. |
| GetNum# | Gets valid numeric input from user | User has input a numerical value and hit enter. | Number is returned to the caller of the function. |
| DoShot | Controls banana shots by accepting player input and plotting | It is a player’s turn. | The players turn is over. |
| ExplodeGorilla | Causes gorilla explosion when a direct hit occurs | A gorilla was hit at a given X,Y position | The gorilla that was hit with a banana explodes |
| Getn# | UNUSED | UNUSED | UNUSED |
| PlotShot | Plots banana shot across the screen | A player or bot has just thrown a banana. | The banana has hit something or gone off screen. The turn is over. |
| CalcDelay! | Calculates the speed at which the computer is running the game. | System is in startup. | System accommodates for computers of different power. |
| InitVars | Sets up the game for the optimal graphics setting, screen size, and game speed. | System is in startup. | Screen dimensions and game information are set. |
| ScreenModeError | prints out an error message about an unsupported graphics color | System is in startup. Integrity of the display is unknown. | Potential display errors have been flagged. |
| PaletteError | Sets mode to 1 | System is in startup. Integrity of the palette is unknown. | Potential color errors have been flagged. |
| DrawGorilla | Draws the Gorilla in either CGA or EGA mode and saves the graphics data in an array. | Gorilla has moved/animated and needs to be redrawn. | An updated gorilla image is drawn to the screen |
| UpdateScores | Updates players' scores | A game has just ended | The winner of the previous game has their score incremented by one |

**2.4 Software Metrics**

|  |  |
| --- | --- |
| Metric | Numerical Result |
| Lines of Code (without the space lines) | 979 ± 78 |
| Commented lines | 198 ± 39 |
| Executable lines | 781 ± 39 |
| Functions/Subroutines | 24 |
| Source Code Files | 1 |
| External Data Files | 0 |