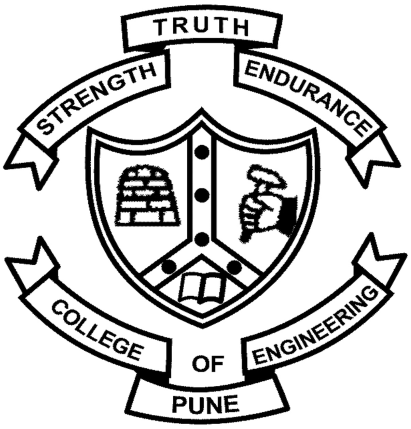
Mini-Project

StealthRun

Submitted by

PIYUSH YADAV (111408069)
SHUBHAM SINGH (111408056)



Department of Information Technology

College of Engineering Pune

Mrs.Rohini Sarode	Prof.Vandana Inamdar	External
Project-Guide	Head of Department	Project In-Charge

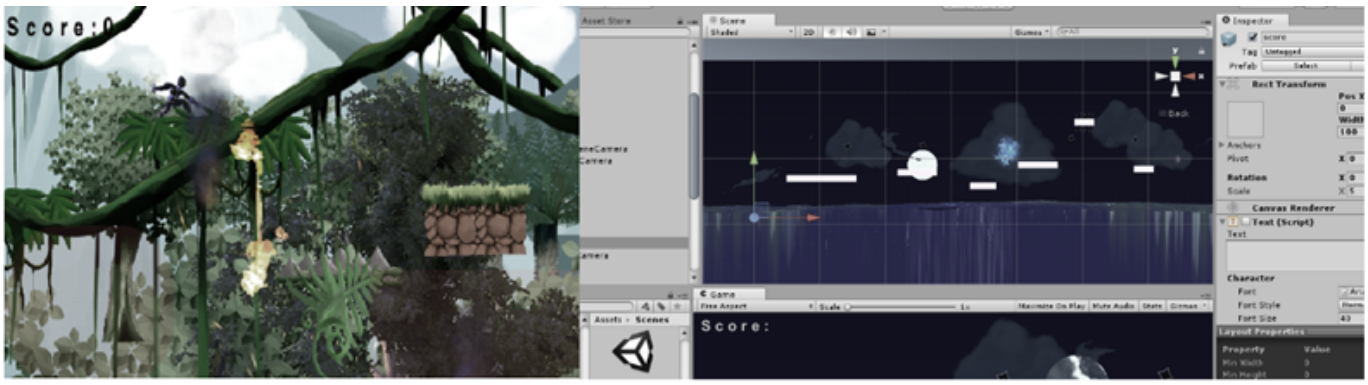
CONTENTS:

ABSTRACT:

Unity is the ultimate game development platform to develop high-quality 3D and 2D games, deploy them across mobile, desktop, VR/AR, consoles or the Web, and connect with loyal and enthusiastic players and customers.

Levels are the space where a player explores the rules and mechanics of a game. As such, good level design is critical to the game design process. While there are many broad design principles, level design is inherently genre-specific due to the wide variety of rules and types of challenge found between genres.

Determining genre specific design principles requires an in-depth analysis of games within the genre. We present such an analysis for the 3D platform genre, examining level components and structure with a better view and understanding their level design. We then use this analysis to present a model for platform levels, specifically focusing on areas of challenge. Our framework provides a common vocabulary for these items and provides us with a method for thinking about elements of platforms and how to compose them to create interesting and challenging levels.



STEALTHRUN

1

INTRODUCTION

Introduction: The mini project which we have developed is a game (named **StealthRun**). The following game is a type of SideScroller game. This game is designed using a gaming engine "**Unity**". This game is solely designed for fun and entertainment purpose.

StealthRun is a game in which the gameplay action is viewed from a side-view camera angle, and the onscreen characters generally move from the left side of the screen to the right to meet an objective. These games make use of scrolling computer display technology.

Good level design is vital for creating enjoyable games; levels provide the player with an interactive space and the ability to explore within the context of the game world rules. As a result, level design is a complex task, requiring an understanding of all the components of the game and how to fit them together. Furthermore, the skills required differ by genre: a level for a 3D puzzle platform has very different requirements than a level for a role-playing game. Despite the different skill sets and the complexity of the task, few texts currently address issues in genre-specific level design. Those that do provide only a brief overview for each genre.

2

LITERATURE REVIEW

Comparison among 3 Side Scrolling types of Games.

"Super Mario" and "**Jump Bug"** are already developed games. The Mini Project game which we have developed is "**Stealth Run"**

Features	Super Mario	Jump Bug	Stealth Run
Total Obstacles	More	More	Medium
Graphics quality	Excellent	Average	Excellent
Portable	Yes	Yes	No
Jump and Run	Yes	Yes	Yes
Shooting	Yes	Yes	No
Various	Yes	Yes	No

Versions			
Drive Car	No	Yes	No
Kill Enemies	Yes	Yes	No
Checkpoint	No	No	Yes

3

REQUIREMENT SPECIFICATION

SRS:

A **Software Requirements Specification** (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

Functional Requirements:

1) Side-scrolling Platform games are action games that feature jumping, climbing, and running characters who must be guided through many diverse levels.

In this game style the player usually starts with a basic platform that flies from left to right and acquires Power-ups that allow them to face an ever increasing horde of obstacles.

2) With games that use side-scrolling, often the screen will scroll forward following the speed and direction of the player character, and can also scroll backwards to previously visited parts of a stage. In such games the screen will follow the player character and scroll only in forward directional, not backwards, so once something has passed off the back of the screen it can no longer be visited. Such types of games have stages where the screen scrolls forward by itself at a steady rate, and the player must keep up with the screen, attempting to avoid obstacles and collect things before they pass off screen.

3) Also the screen in many games that use side-scrolling, for the most part, follows the player character and tries to keep it near the center of the screen.

Non Functional Requirements:

Some non functional requirements for project are:

1) Accessibility

2) Effectiveness

3) Response time

4) Stability

5) Fault tolerance

6) Portability

7) Reliability

8) Testability

Software Requirements:

1) First of all we need to install Unity Game Engine from the official website of Unity in Windows or Mac OS.

2) Programming languages such as C#, Java, Python or any high level language can be used to develop the game.

3) Also good knowledge of Physics topics such as collisions, kinematics, rotational motion and motion in a plane can be useful to develop a creative game.

4) Knowledge of Mathematical formulas and concepts of geometry can also be an additional bonus for a developer.

Hardware Requirements:

1) OS: First of all the operating system must Windows 7 SP1+, 8, 10; Mac OS X 10.8+. Windows XP & Vista are not supported and server versions of Windows & Mac OS are not tested.

2) GPU: Graphics card with DX9 (shader model 3.0) or DX11 with feature level 9.3 capabilities. The rest mostly depends on the complexity of projects.

3) CPU: SSE2 instruction set support.

Constraints:

Like any tool, it has strengths but also some limitations.

1) The most obvious limitation is that it doesn't allow us to start from a foundation, or a template. We need to implement all the details. We have to start from scratch with each game. "As a general-purpose engine, it does not offer any base or model. Some people develop games and expect it to be all "drag and drop" but it is not like that.

2) From a graphical point of view, it is also lagging behind compared to other engines like UDK. While in contrast, Unity 3D allows programming 'shaders' from scratch.

3) The fact that there are more expensive licenses can be a limitation for freelance developers and small development groups. The most expensive licenses provide mainly graphical and performance improvements, but it is only worth it if you have medium or large equipment, and an ambitious project.

4

PROPOSED SYSTEM

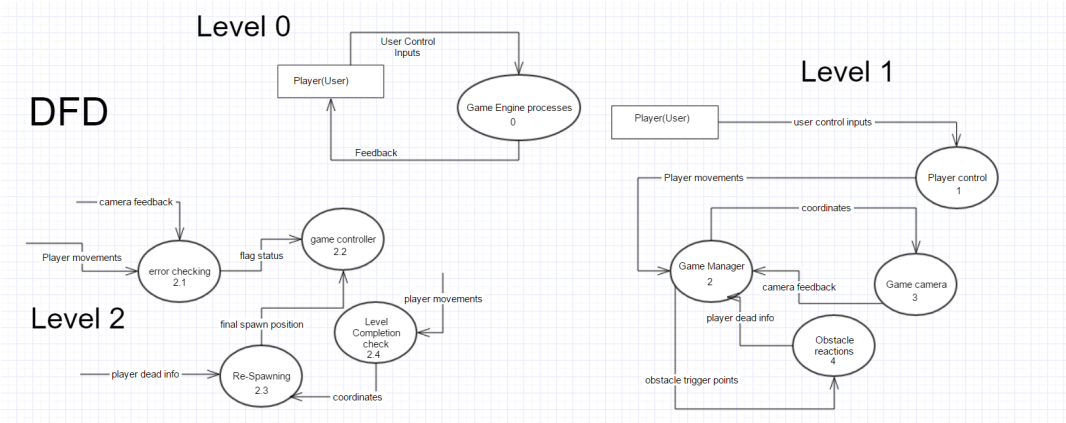
Problem Statement: Our Aim is to design a SideScroller Game on Unity Gaming Engine. The game is developed for fun and entertainment purpose only.

Problem Definition: Side-scrolling Platform games are action games that feature jumping, climbing, and running characters who must be guided through many diverse levels.

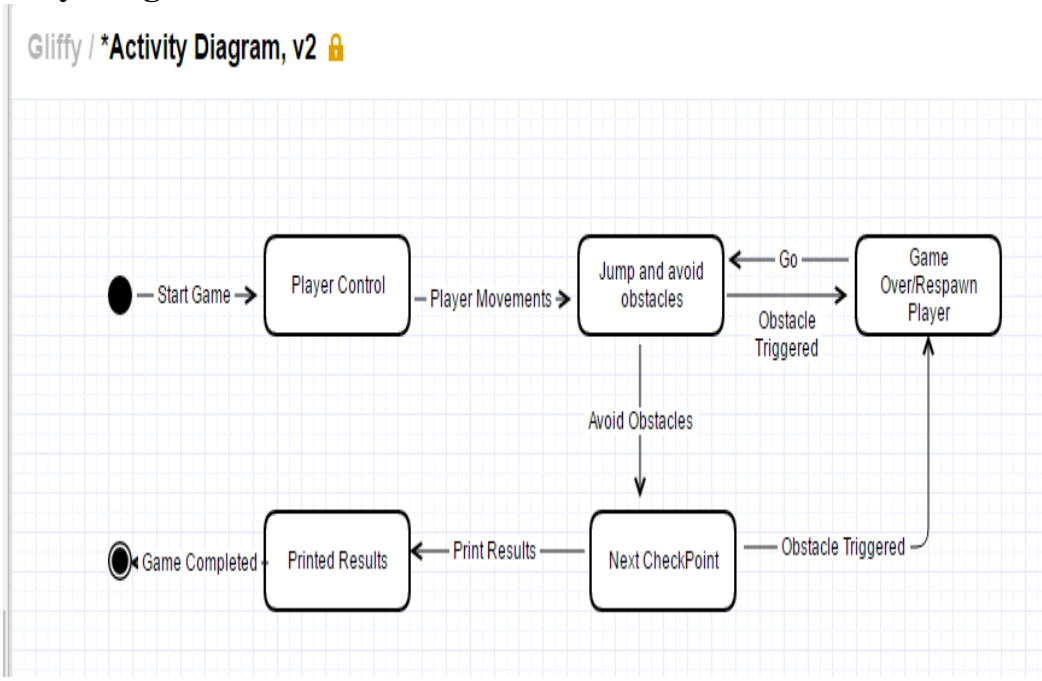
With games that use side-scrolling, often the screen will scroll forward following the speed and direction of the player character, and can also scroll backwards to previously visited parts of a stage. In such games the screen will follow the player character and scroll only in forward direction, not backwards, so once something has passed off the back of the screen it can no longer be visited. Such types of games have stages where the screen scrolls forward by itself at a steady rate, and the player must keep up with the screen, attempting to avoid obstacles and collect things before they pass off screen.

Design :

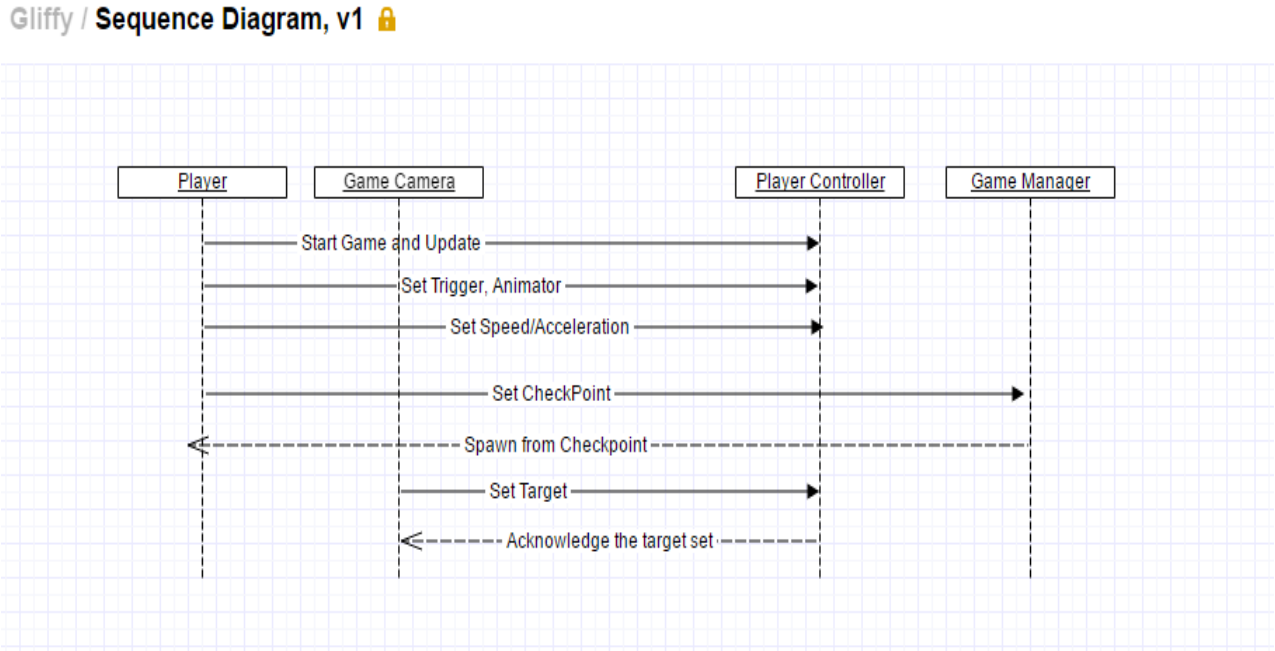
DFD Model :



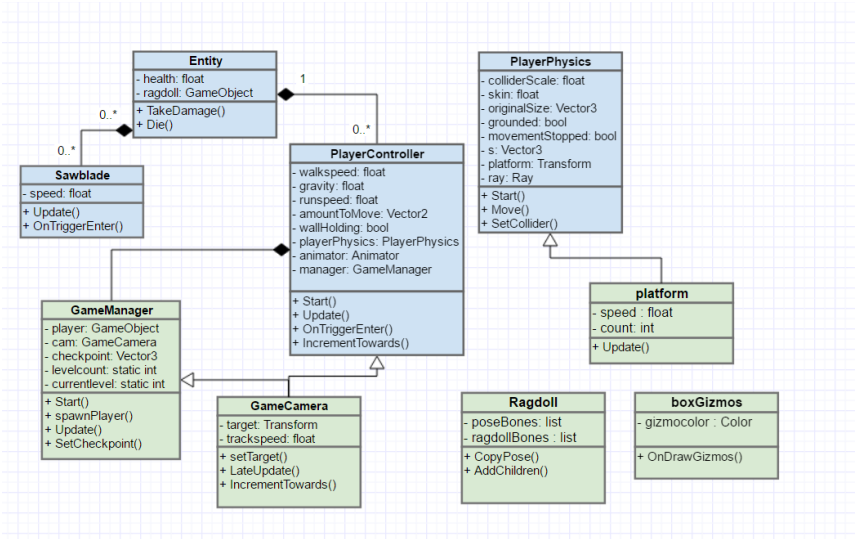
Activity Diagram :



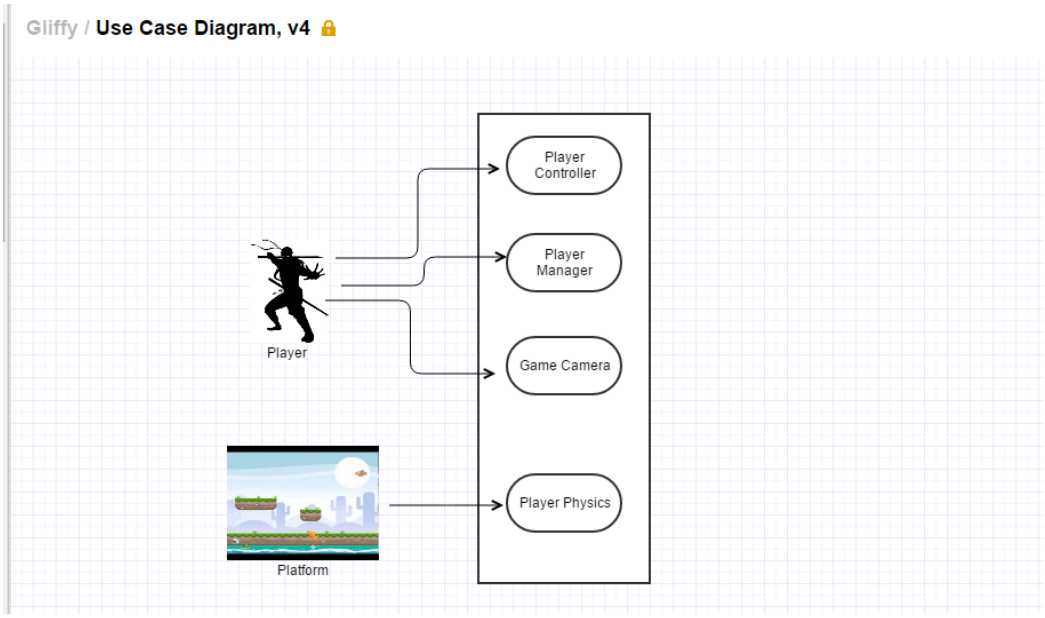
Sequence Diagram :



Class Diagram :



Use Case:

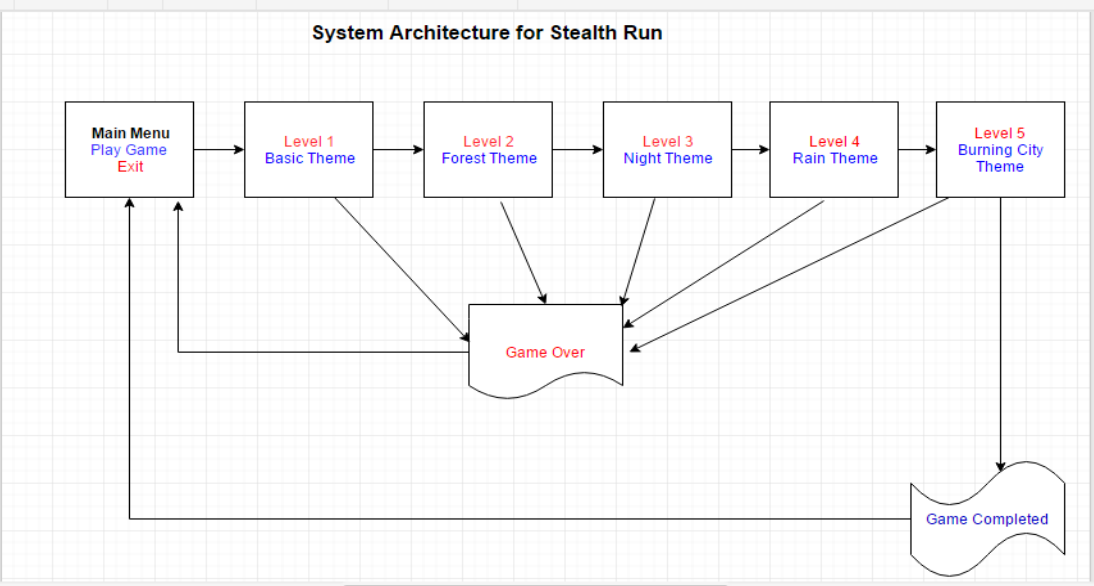


MODULE DESCRIPTION:

System Architecture:

A System Architecture is a conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

A system architecture can comprise system components that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture; collectively these are called architecture description languages (ADLs).



Modules:

Player Controller:

This module is handled by a script which manages the controls of the player(ninja). The collision and acceleration along with the jumping and

walking of the player are handled in this module. The total score is calculated in this module according to the input received from the coin class.

Player Physics:

All the physics and calculations related to the player and the obstacles are handled in this module. The vertical and horizontal collisions are are calculated and the result is feeded to the Player Controller module. The colliders of the moving platforms are also set in this module.

Game Manager:

This module is like the heart of the GUI as well as the game. The score that is displayed continuously on the screen is refreshed and updated by this module and its scripts. The control of different levels i.e. changing of levels when reaches a checkpoint is done here as well. The spawning of player and the death count for GameOver script to run is managed too.

Game Camera:

Since, it's a sidescroller game, the very important part of camera movement along with the player is handled by this module. The distance to be moved and the acceleration are all calculated with reference to the player movements and then the camera position is updated accordingly.

Death:

Whenever a player falls on a spike or goes into the fire or falls off the surface(Infinite Fall), the death module and script are triggered which takes care of the death if the player and accordingly the lives left are updated and the player is respawned.

Menu:

The Main Menu is designed and managed by this module. The canvas and the buttons inclusive of the functions that are triggered by each button can also be managed by this module.

5

PROJECT SCHEDULE AND ESTIMATION

Lines of Code:

LOC is the simplest among all metrics available to estimate project size. Project size is estimated by counting the number of source instructions in the developed program.

Obviously, while counting the number of source instructions, lines used for commenting the code and the header lines should be ignored.

Game Scripts:

File Name and No. of Lines. .cs means that the code is written C sharp language.

PlayerPhysics.cs: 145
GameCamera.cs : 33
GameManager.cs : 100
Entity.cs : 30
platformvert.cs : 20
holdCharacter.cs : 14
platformrotate.cs : 12
death.cs : 13
PlayerController.cs : 147
Coin.cs : 31
Platform.cs : 20
SawBlade.cs : 25
Menu.cs : 32
Boxgizmos.cs : 18
testmovmtX.cs : 30
testmovmtY.cs : 30

Water Scripts :
Display.cs : 36
GernsterDisplay.cs : 9
MeshContainter.cs : 27
PlanarReflection.cs : 284
SpectacularLightning.cs : 33
Water.cs : 403
WaterBase.cs : 78
WaterTile.cs: 67

Cars Scripts:
TakeScreenShot.cs : 22

Warrior Pack Bundle:
IKHands.cs : 36
SmoothFollow.cs : 91
WarriorAnimationDemo.cs : 116

Total Lines: 1902

Other than Code various other Models, Materials, Textures, Prefabs and Animations are used to develop the game.

COCOMO Model:

- This model estimates the total effort in terms of "person-months" of the technical project staff.
- Boehm introduces three forms of cocomo. It can be applied in three classes of software project:

    1. Organic mode
    2. Semidetached mode:
    3. Embedded mode:

Organics Model is relatively simple, small projects with a small team are handled. Such a team should have good application experience to less rigid requirements.

Since the model we have designed takes into consideration these facts, our COCOMO Model is an Organic Model.

The basic cocomo model takes the following form:

Where KLOC is the estimated size of the software product expressed in Kilo Lines of Code, a1, a2, b1, b2 are constants for each category of software products, Tdev is the estimated time to develop the software, expressed in months, Effort is the total effort required to develop the software product, expressed in person months (PMs).From Effort & Tdev we can compute the no: of people required to accomplish the project as N=E/D

$$\text{Effort} = a_1 \times (\text{KLOC})^{a_2} \text{ PM}$$

$$\text{Tdev} = b_1 \times (\text{Effort})^{b_2} \text{ Months}$$

Cocomo Organic Model Table:

Model	a1	a2	b1	b2
Organic	2.4	1.05	2.5	0.38

In Our Model Total Number of Lines of Code is: 1902.

So Total No. of Kilo Lines of Code: 1.902

1. Effort Applied (E) = $2.4*(1.902)^{1.05}$

$= 4.7140$

2. Development Time (Tdev) = $2.5*(4.7140)^{0.38}$

$= 4.50$

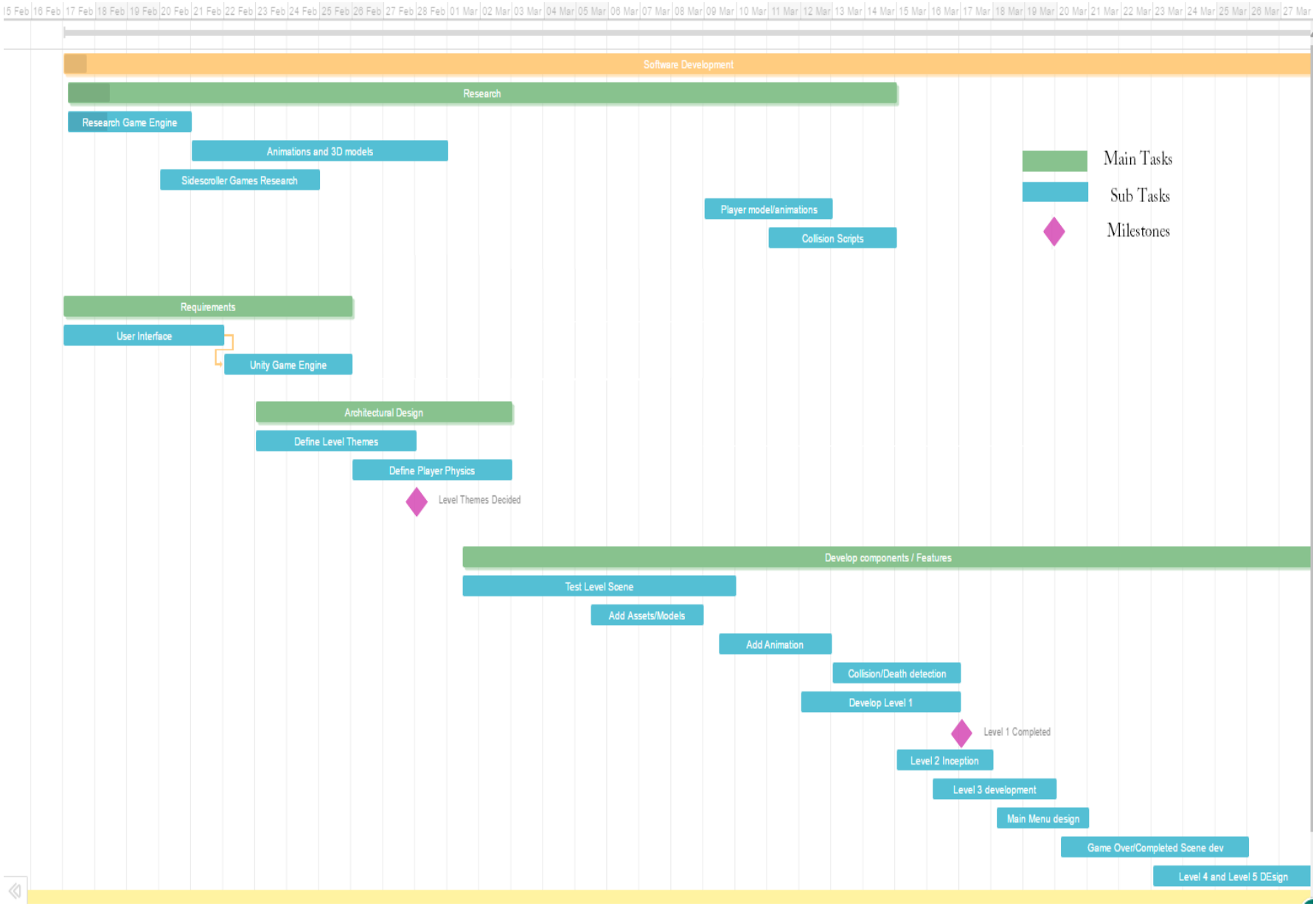
3. People Required = Effort Time/Development Time

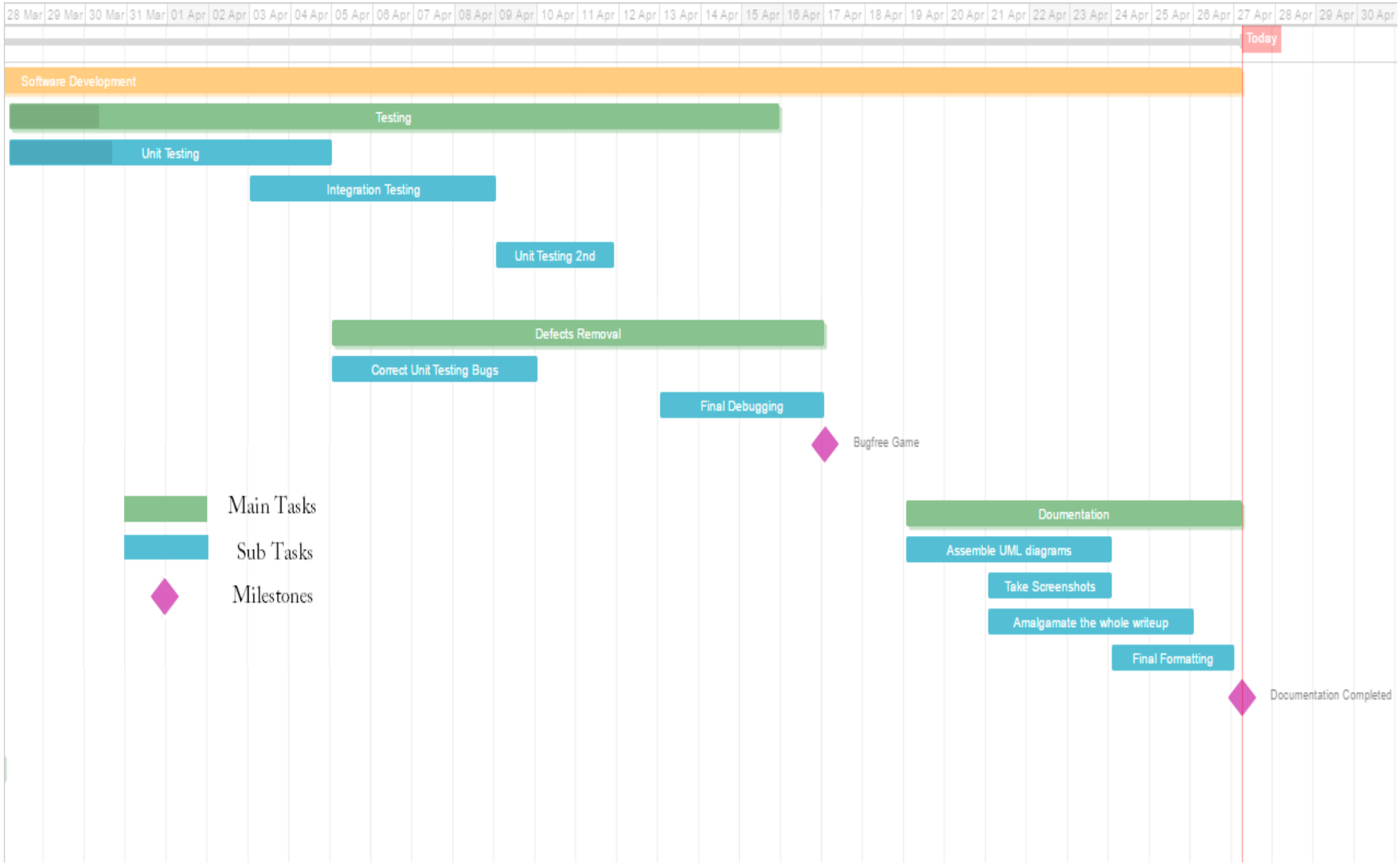
$= 4.7140/4.50$

$= 1.05$

Other than the code – There are various Models, Animations, and Textures which has also been included in our Unity Game Engine which adds more contribution to our model.

Gantt Chart for StealthRun

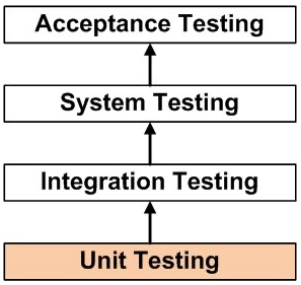



28 Mar
29 Mar
30 Mar
31 Mar
01 Apr
02 Apr
03 Apr
04 Apr
05 Apr
06 Apr
07 Apr
08 Apr
09 Apr
10 Apr
11 Apr
12 Apr
13 Apr
14 Apr
15 Apr
16 Apr
17 Apr
18 Apr
19 Apr
20 Apr
21 Apr
22 Apr
23 Apr
24 Apr
25 Apr
26 Apr
27 Apr
28 Apr
29 Apr
30 Apr
Today
Software Development
Testing
Unit Testing
Integration Testing
Unit Testing 2nd
Defects Removal
Correct Unit Testing Bugs
Final Debugging
Bugfree Game
Main Tasks
Sub Tasks
Milestones
Doumentation
Assemble UML diagrams
Take Screenshots
Amalgamate the whole writeup
Final Formatting
Documentation Completed

6

TESTING AND RESULTS

Testing:



Unit Testing:

Unit testing is a level of software testing where individual units/ components of software are tested. The purpose is to validate that each unit of the software performs as designed.

Test Cases	Test Case Name	Expected Result	Actual Result	Status
1	Jump and Move	Jumping over Platforms and Obstacles	Jumping over Platforms and Obstacles	Passed
2	Saw Blade	Take Damage and Destroy Player. Play Audio	Take Damage and Destroy Player. Play Audio	Passed
3	Fire	Take Damage and Destroy Player	Take Damage and Destroy Player	Passed
4	Spikes	Take Damage and Destroy Player. Blood Animation	Take Damage and Destroy Player. Blood Animation	Passed

5	Vertical Platforms	Avoid Collisions Move Up & Down	Avoid Collisions Move Up & Down	Passed
6	Static Platforms	Avoid Collisions	Avoid Collisions	Passed
7	Infinity Fall	Take Damage. Die	Take Damage. Die	Passed
8	Coins	Collect Coin Score = +1	Collect Coin Score = +1	Passed
9	Game Over	Die. Game Over Scene	Die. Game Over Scene	Passed
10	Game Completed	Complete Last Level. Game Completed Scene	Complete Last Level. Game Completed Scene	Passed

Integration Testing:

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Objective is to take unit tested modules and build a program structure based on the prescribed design

Two Approaches

1) Non-incremental Integration Testing
2) Incremental Integration Testing

Test Case	Test Case Name	Expected Result	Actual Result	Status
1	Level Switching	Transition from One l Another	Transition from One l Another	Passed
2	Game Over to Main	Transition from Gam to Main Menu	Transition from Gam to Main Menu	Passed
3	Main Menu Options	Play Game or Exit	Play Game or Exit	Passed

7

CONCLUSION AND FUTURE SCOPE:

With a formal definition of a SideScroller Game, including a model for platform-based challenge, there are a number of different kinds of analysis we can perform. For example, it may be helpful to look at ways of classifying the difficulty of a level based game on the pace of a platform and length of a level.

A short section of a stage may be difficult to master and therefore provide more difficulty to the player. Similarly, an extremely long and rhythmic grouping may be easier at first but may require the player to concentrate longer, leading to more points of potential failure. We can set various constraints or obstacles and add more levels to this game to make it more realistic.

This framework can be especially helpful to educators who teach students how to design fun and challenging stages in games. It is our hope that this framework will also provide a common vocabulary for both game analysts and developers and open the field to similar frameworks for other genres.

Also, we are looking to integrate our game with Android, ios and Windows mobile OS. 80% of the Android work is completed. We are looking forward to change the player controls and add more level, animations, enemies, twists and better themes.

References:

1) Research papers:

a) A Framework of Analysis of 2D Platform Levels - Gillian Smith, Mee Cha, Jim Whitehead

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.494.4080&rep=rep1&type=pdf

b) Rhythm Based Level Generation for 2D Platformers - Gillian Smith, Mike Treanor, Jim Whitehead, Michael Mateas
https://users.soe.ucsc.edu/~ejw/papers/smith-platformer-generation-fdg2009.pdf

2) Asset Store Link: https://www.assetstore.unity3d.com/en/#!/

3) YouTube Series Video of Sebastian Lague of SideScroller Game
https://www.youtube.com/watch?v=d3HEFiDFApI&list=PLFt_AvWsXl0e-g21S-MiPArfEhlv9pUgC

4) SideScroller Wikipedia page

https://en.wikipedia.org/wiki/Side-scrolling_video_game

5) 3D Models

https://www.turbosquid.com/Search/3D-Models/free

https://www.mixamo.com/store/#/

https://www.yobi3d.com/q/spikes?page=1

https://opengameart.org/content/spikes-1

https://www.3dbuzz.com/training/view/unity-fundamentals/physics/15-cloth-flag