Rollout!

Game Functional Requirements

Creator: Connor Igo

Email: igo.co@husky.neu.edu

**Rollout! Game**

Rollout! Is a take on an endless runner game where instead of running along a course the player rolls along, dodging obstacles and avoiding crashing or falling off the path. Speed boosts and jumps will allow players to traverse the course quicker, where objects and other items will attempt to impede a players movement. This is similar to a typical runner except the player is expected to stay on course, jumping into a turn will send the player flying off the edge to their death instead of keeping them on track!

The image below is from <https://wiki.unrealengine.com/>

****

The image below is from <http://www.wowhead.com/> with a mini-game of very similar idea

****

**Contents**

1 Requirements Analysis 2

2 Physical Architecture Analysis 3

2.1 System Architecture 3

2.2 Application Layers 3

2.2.1 Software Requirements 3

2.2.2 Hardware Requirements 3

2.2.3 Data Access 3

2.3 Application Platform 4

2.3.1 Administrative Tools 4

2.4 External Systems & services 4

2.4.1 Email Service 4

2.5 Communication Protocols 4

2.5.1 HTTP 4

2.5.2 OpenSSL 5

2.6 Database Model 5

2.7 Hardware & Software Infrastructure 5

2.7.1 Standard deployment procedure 5

3 Programming Analysis 7

4 Component Analysis 7

5 Evaluation 9

6 Milestones 9

7 References 10

8 Contract Modifications 10

# Requirements Analysis

Rollout! Is a desktop computer game.

Game Scoring:

* Keep track of distance
* Keep track of items collected
* Keep track of time
* Keep track of previous records

Character Movement :

* Rolling animation
* Character movements satisfy basic physics law
* Jumping follows appropriate physics

Game Controller:

* Keyboard
* Mouse

Level Builder:

* Barriers
* Speed Boosts
* Speed Slows
* Coins
* Terrain

Rules:

1. Race to beat previous scores, avoiding barriers and falling off the course

# Physical Architecture Analysis

Rollout! is composed of several technologies that stack upon each other to compose the architecture that enables all the required services and provides the user experience.

All client‐side technologies are based on a multiplatform build supported by Unreal Engine, using C++ backed code. Any server-side technologies would be integrated through Valve pending a Steam launch.

## System Architecture

The system is based on a client-side desktop application.

Client Side

* Windows
* OS X
* Linux

## Application Layers

Rollout! uses Steam for friend data as well as networking on the client side.

### Software Requirements

The software:

* Windows
* OS X
* Linux
* Xcode
* C++

### Hardware Requirements

The hardware:

* Any computer running an x32 processor and at least 4GB of RAM

### Data Access

Pending, will use available Steam APIs for Data.

## Application Platform

### Administrative Tools

#### Unreal Scripts

Rollout! will have a set of administrative tools written in C++ and python. Currently the following tools are included in the admin:

* Statistics
* Start/Stop game server daemon

## External Systems & services

### Email Service

All Application emails are sent by the system using a batch process.

The process sends all emails on each run using the SMTP mail protocol.

Initially, the SMTP server resides within the EC2 instance, but can be separated as server load grows.

## Communication Protocols

All communication protocols are pending integration through a release on Steam, utilizing APIs provided by Valve.

### HTTP

Hypertext Transfer Protocol (HTTP) is the method used to transfer or convey information on the World Wide Web. It is a patented open internet protocol whose original purpose was to provide a way to publish and receive HTML pages.

### OpenSSL

The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography.

## Database Model

Any game data will be stored locally using SQL databases, pending online storage through Valve.

## Hardware & Software Infrastructure

Rollout! is currently planned to be hosted in Amazon EC2. Both environments (staging & production) are running on the same single instance.

### Standard deployment procedure

This section describes the installation of the Rollout! system on Amazon EC2; it is intended as a guide for a minimal system upon which further configurations can be applied.

#### Game server ports

We're setting all the default ports for OpenSSL, those can be changed but you'll probably need at least access to the web admin panel unless you connect locally from inside the instance.

OpenSSL default ports are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Interface | Port |  | Type | Description |
| All addresses | 5555 |  | Client to Server | The port for clients to connect to the server using OpenSSL. Connections may or may not be encrypted. You can update the security settings for this port. |
| All addresses | 80 |  | HTTP Binding | The port used for unsecured HTTP client connections. |
| All addresses | 443 | S | HTTP Binding | The port used for secured HTTP client connections. |

Allow all those as TCP connections.

# Programming Analysis

Connor Igo will be responsible for all programming of Rollout!.

# Component Analysis

Game Class:

The Game class will keep track of:

* Game Time
* Score
* Previous Scores

The Game class will allow:

* Time display
* Score display
* Game Difficulty (optional)
* Parental Controls (turn off death animations / blood)
* Spawning Objects
* Spawning the Player
* Defeat or new high score
* Game Boundaries (path boundaries)

Pickup Class

The Pickup class will keep track of:

* Type of pickup
* Pickup location
* Pickup size

The Pickup class will:

* Handle Pickup display
* Handle Pickup collision
  + Player

Terrain Class

The Terrain class will keep track of:

* Terrain Position

The Terrain class will allow:

* Collision with player

Player Controller

The Player Controller class will keep track of:

* Input from the keyboard/mouse

The Player Controller class will:

* Move Player based on input from the keyboard/mouse
* Movement
  + Moving outside of allowed area triggers a loss

Death Class

The Death class will keep track of:

* Death animation

# Evaluation

The following tests will be used to evaluate the quality of the software:

Unit Tests

* A positive assertion for all class methods
* A negative assertion for all class methods

Play Tests

* Connor will play/user test any component with user input

Simulations

* Data will be simulated when missing or sparse

Security tests

* Automated stress and password hacking will profile each component (pending network integration)

Metric Logs

* Play statistics will be logged

Dynamic Profiling

* Valrind will be used to profile all components

# Milestones

The following milestones will be due at the following dates:

March 4th

* Basic runner game complete, some models as well for terrain / boosts. Stretch is to have player rolling instead of running.

March 25th

* Have at least infinite level with components generating, stretch is to have collectables as well.

April 15th

* Prototype of game, terrains, pickups and models working.

# References

Rollout! is composed of several application components. It is written using C++ in Unreal. Server side applications pending.

References (including potential resources):

* Apache: <http://httpd.apache.org/>
* Python: <http://www.python.org/>
* Ubuntu Linux: <http://www.ubuntu.com/>
* Django: <https://www.djangoproject.com/>
* Amazon EC2: <http://aws.amazon.com/ec2/>
* mod\_wsgi: <http://code.google.com/p/modwsgi/>
* Cocos2d-x: <http://www.cocos2d-x.org/wiki/Cocos2d-x>
* OpenSSL: <http://www.openssl.org/>
* Boost C++: <http://www.boost.org/>
* Boost.ASIO: <http://www.boost.org/doc/libs/1_54_0/doc/html/boost_asio.html>
* Xcode: <https://developer.apple.com/xcode/>

# Contract Modifications

Any changes to this document will be e-mailed to the professor with the changes highlighted and the date. The professor must approve of the most recent changes,