Conundrums

Technical Design Document (TDD)

Group 3:
Danise Edra
Peter Hong
Justin Olaso
Michael Zimmerman

'Quite the Conundrum'

Table of Contents

Game Development Team Members

PRODUCER
Danise Edra
Peter Hong
Justin Olaso
Michael Zimmerman

PRODUCTION MANAGER
Danise Edra
Peter Hong
Justin Olaso
Michael Zimmerman

PRODUCTION COORDINATOR
Danise Edra
Peter Hong
Justin Olaso
Michael Zimmerman

GAME DESIGNERS
Danise Edra
Peter Hong
Justin Olaso
Michael Zimmerman

SYSTEMS/IT COORDINATOR
Danise Edra
Peter Hong
Justin Olaso
Michael Zimmerman

PROGRAMMERS
Danise Edra
Peter Hong
Justin Olaso
Michael Zimmerman

TECHNICAL ARTISTS
Danise Edra
Peter Hong
Justin Olaso
Michael Zimmerman

AUDIO ENGINEERS
Justin Olaso

UX TESTERS
Danise Edra
Peter Hong
Justin Olaso
Michael Zimmerman

Executive Summary

Game Overview

Title: Conundrums
Platform: PC Standalone
Genre: RPG Adventure, Party, Board game
Rating: (10+) ESRB
Target: Casual gamer (aging from 10 - 30)
Release date: May 11, 2017
Publisher: DJ PM

Conundrums is a third person boardgame/minigame game where the player race to the top and try to have the highest score. As they move through the board minigames are triggered and the players compete to win the points. When someone reaches the top, the person with the highest score wins.

Technical Summary

Conundrums is to be developed over a couple of months by a team of 4 people using the Unity Game Engine. The player character art will be created by one of the developers and other art and music assets will be sourced from the store. The player character will be made using Spriter and the Spriter2Unity plugin. The game will be built for Windows. The game will cost zero dollars and the game is expected to make nothing in revenue. It is free to play.
Hardware requirements: If it can run the Unity editor, then it can probably run the game.

Equipment

The production of this game will have the developers using their own devices. Also using our own controllers for production. The game is up to four players. Player one uses the keyboard and the other 3 will need a gamepad to operate.

Hardware

Each member will utilize their own laptop as their primary hardware platform for game development. Additional hardware utilized are standard game controllers.

Software

Unity Pro - Education License Free
Spriter Pro - $3
Google Docs
Krita
GitLab

Evaluation

Game Engine

The game engine utilized by all team members for the development of *Conundrums* is Unity for its ease of use and availability.

Target Platform

Conundrums is deployed to the PC platform.

Scheduling

Development Plan

Development of *Conundrums* will be done via version control through Git on the Rijeka server. Each team member will create their own working branch. Work gets pushed to the “staging” branch, and when a working project mark has been reached, “staging” will get pushed to “master.”

Milestones

- Generate game board procedurally
- Create player Sprites
- Create several minigames
- Implement scoring for players
- GUI, menus

Work Environment

Remote Collaboration

The team consists of four people. Each team member will be developing scenes for *Conundrums* remotely, and version control will be done through Git on the Rijeka server. The team will make use of the “issues” section in order to address problems faced when developing *Conundrums*.

Game Scripts

Board_Generator - This generates the board at start and has functions for the players to call to fill in the board as they traverse the scene. The board will also generate other decorations onto the board and will generate ladders to go up another level.

void Start() - O(1) only initializes some variables

void board_setup() - O(columns*row) the columns and row are to be set from the inspector. Generates initial tiles rows(-5 to row) and columns(-5 to columns). By default we have it to generate 10 by 10 board for 100 tiles. It makes the tiles alternate from black and white tiles. That function that checks adjacent tiles runs in constant time.

void add_tiles() - O(1) the function checks if a tile it is told to add already exists, if it doesn't it adds it, if it doesn't the function ends

void add_to_board() - O(1) it always passes to add_tiles() 9 tiles to try and add to the board

new_floor() - O(columns*row) same as board_setup() but as a public function that takes into account what floor the player should be in and is to be called by players that collide with a ladder

bool black_adjacent() - O(1) always check four edge adjacent tiles. If any of them are black return true so the other functions knows to set it to white or black

DeathByPlinkoController - Scene controller. Spawns the player objects and handles the winning
- Start() - O(number of players) some number 1-4
- SpawnObject() - O(1) does one action each time
- setText() - O(1)
- winner() - O(1)

PlinkoRolling
- Start() - O(1)
- OnCollisionEnter() - O(1)

StartPlinkoMiniGame
- Start() - O(1)
- StartGame() - O(1)

SwordSmack
- OnCollisionEnter() - O(1)

DontTouchTheLavaController
- Start() - O(num of players)
- Update() - O(1)
- setText() - O(1)
- winner() - O(1)

PlatformDescend
- Start() - O(1)
- BeginMiniGame() O(1)
- Update() - O(1)
- ChooseNextTile() -

StartDontTouchTheLavaMiniGame
    Start()
    StartGame()

FallenSky
    Start()

FallenSkyController
    Start()
    Update()
    SpawnObject()
    setText()
    winner()

FollowPlayer
    Start()
    LateUpdate()

PauseMenuFS
    Start()
    Update()
    PauseGame()
    ClickedButton()
    resetPanels()

PlayerController
    Start()
    Update()
    movePlayer()
    TakeDamage()
    OnCollisionEnter()
    flip()
    swordSwing()

StartMiniGame
    Start()
    StartGame()

Roll_Manager
    Start()
    update_points()

Update()

Rolling
- Start()
- FixedUpdate()
- OnCollisionEnter()
- OnTriggerEnter()

Dice
- Start()
- Update()
- OnTriggerEnter()
- step_made()

Game_Manager
- Awake() - O(1)
- Start() - O(number of players)
- Update() - O(1)
- pick_minigame() - O(1)
- return_from_minigame() - O(number of players)
- store_player_state() - O(number of players)
- setAllActive() - O(number of players)

GameFinished
- finished()
- whoWon()
- nnumPlayerWinMas()

LoadSceneOnClick
- LoadByIndex()

NumPlayers
- Awake()
- setNumberOfPlayers()
- setNumberofFloors()

PauseMenu
- Start()
- Update()
- ClickedButton()

Player
- Awake() - O(1)

Start() - O(1)
Update() - O(1)
FixedUpdate() - O(1)
movement() - O(1)
OnTriggerEnter - O(1)
forward_is_clear - O(1)
set_player_num - O(1)
get_turn - O(1)
set_turn - O(1)
set_board_manager - O(1)
setPlayerColor - O(number of armor)

PlayerScores
- Start()
- Update()
- scoreText()

QuitOnClick
- Quit()

ReturnToMainMenu
- CleanSlate()

SelectOnInput
- Start()
- Update()
- OnDisable

SliderValues
- Start()
- SubmitSliderSetting

VolumeManager
- Awake()
- Update()
- UpdateVolume()

VolumeManagerSpawner
- Start
- ManageVolume