Diavolo 3 Technical Design Document

Derivative Games

Overview

Diavolo 3 is an isometric hack'n'slash action RPG. The player chooses a character from one of several classes (called Archetypes) and levels them up by defeating enemies and doing quests. The character collects Equipment along the way to further boost their abilities.

There will be 4 Acts, each created as a single scene. The vertical slice will contain Act 1

It will be developed in Unity 2020.3

No major third party add-ons are required.

Game Management and Scenes

The following scenes will be in the project

- Main Menu
- Game HUD
- Act1
- Act2
- etc

The Main Menu is loaded at startup.

GameManager

A GameManager singleton will be built into the Main Menu scene, and is made immortal via DontDestroyOnLoad.

The GameManager clones a player character based on the prefab selected in the main menu (depending on the character's archetype) and is responsible for communication between this main player character and the in game HUD and other elements that need to know about the player.

Each Act of the game is developed as a separate scene with its own spawners, quests and so on built in as well as the level geometry.

The gameplay elements for each Act (spawners etc) will be kept as a prefab to allow a gameplay designer to work on them while an artist or level designer works on the level geometry, to minimise source control conflicts.

The Game HUD is a separate scene that is loaded on top of each scene via additive scene loading.

Character Components

Character

The Character component contains the current state of the RPG Character

- Maximum and current Health and Mana, a list of status effects
- Level, class, XP, and list of available Skills
- Dictionary of stat values (eg speed multiplier, damage multiplier, damage resistance, dodge, hold/stun values) that can be affected by Buffs
- Equipment worn by the characters and references to the Status effects per slot

CharacterMover

Controls character animation and states like knockback. Communicates with the Animator, NavMeshAgent and CharacterController components

AIController

Determines where the character should move. Communicates this to the CharacterMover

PlayerController

Determines where the character should move to based on mouse clicks, and activates powers based on keypresses or other UI. Communicates this to the CharacterMover and Character

AIBrain

A simple Utility AI that evaluates an NPC's powers and uses them. Can also incorporate other Actions such as Patrolling or Wandering

A player controlled character prefab will have the following components:

- Character
- PlayerController
- AIController
- CharacterMover
- CharacterController
- NavMeshAgent
- Animator
- SkinnedMeshRenderer

An NPC will have the following components

- Character
- AIBrain
- AIController
- CharacterMover
- CharacterController
- NavMeshAgent
- Animator
- SkinnedMeshRenderer

Characters (both player and NPC) will be set up on a prefab for each type of character using these components.

RPG Components

Skill

Abstract base class for skills which can affect allies, enemies or self. Contains damage and list of status effects to apply to both caster and target

Also contains an Animation enum (which is turned into a trigger for the Animator) and references to visual effects for start, activation and impact, and an Icon to use in various UIs for the skill.

Skills have a function to create a programmatically created tooltip. This takes the stats from the power and the description for each of its status effects to return a player-facing description

e.g

Firebolt
Projectile: 3-8 Fire Damage, Range 20m
10 Mana, 5 sec cooldown
4 Fire damage/second for 3 seconds

These are assembled using the C# StringBuilder class

Derived classes: SkillMelee, SkillRanged, SkillProjectile, SkillPBAoE, SkillSelf

Status

Abstract base class for effects that are applied by Skills or Equipment. Specifies an icon and particle system for while the status is active.

Status has a virtual function to return a player-facing description, eg.

- *+10% Damage*
- *3 Fire Damage/second*
- *Stun for 6 seconds*

Derived classes: Buff, DamageOverTime, Knockback, Aura, Mez

Archetype

CharacterClass, eg Warrior, Mage, which points to a list of skill trees, and base stats foe health and mana

SkillTree

List of Skills each with a level that they unlock at, and predecessor skills required to unlock them

AIBrain

A central UtilityAI that decides what Skill each NPC should use when they have completed their previous skill action

PlayerBrain

Responds to keyboard input and directs the player character to use their skills when keys are pressed

Equipment

ScriptableObject that specifies which slot the equipment sits in (eg Head, Amulet, Body, Boots) or Consumable.

Has a list of Status effects, which are imbued on the character when they wear it, and removed when they take it off. Consumables confer their status effect with a duration when consumed.

Each piece of Equipment can generate its own tooltip text programmatically based on the status effects it confers e.g

Ruby Helmet of Osiris
Defense Rating 10
+10% Fire Resistance
+20% Perception

Each Equipment scriptable object will reference a Sprite to use for the equipment in the Inventory UI, and a Mesh to use on a standard Equipment piece prefab when the object drops into the game world.

This prefab will be a simple BoxCollider and Rigidbody game object with a trigger SphereCollider for the player to pick it up. The pickup trigger code is in an EquipmentInstance script, which also holds a reference to the Equipment scriptable object that is being represented.

RewardTable

A table of Equipment and an amount of gold that can be dropped by an NPC on defeat or given for completing a Quest. When a RewardTable is rolled (a chest is opened, or an enemy is defeated), a random object is selected from the table to instantiate.

Quest System Components

QuestManager

Singleton component that finds all Quests in the current scene and responds to their completion.

Quest

Abstract base class Component that holds a target number of things to do and a virtual function for getting the current amount.

Quests contain descriptions and a reward table, and are built into the scenes for each Act.

Each quest can have predecessor Quests that must be completed before they unlock.

Quests have associated dialog for unlocking and completion.

Derived classes: QuestDefeat, QuestCollect, QuestLocation

HUD and UI classes

SkillBar and SkillUI

The SkillBar gets a reference to the player from the GameManager and clones a prefab SkillUI for each of the player's Skills. This SkillUI shows the icon, has a radial fill overlay for cooldowns, and has a tooltip to show the name and stats for the skill. Pressing the button activates the skill, as does pressing the appropriate key.

CharacterUI

Can show a character's name, health, mana and icons for active status effects. This script is used for the Overlay HUD health/mana orbs and status effect for the player.

A different prefab is also used for world space canvas health bars above NPC's heads.

QuestManagerUI

Shows a text description and progress bar for each open quest via the QuestManager. References a prefab QuestUI and clones a copy of it for every Quest in the QuestManager

QuestUI

UI for an individual Quest, showing its description and progress

Suggested Folder Structure

Character prefabs and associated powers and scripts are kept together

- Assets/Characters/Player/Mage/Mage.prefab
- Assets/Characters/Player/Mage/Firebolt.asset (Power scriptable object)
- Assets/Characters/Player/Mage/FireboltFlames.asset (Status scriptable object)
- …
- Assets/Characters/Player/Warrior/Warrior.prefab
- …
- Assets/Characters/NPC/Enemies/Goblin/Goblin.prefab
- Assets/Characters/NPC/Enemies/Orc/Orc.prefab
- …
- Assets/Characters/NPC/Helpers/Amazon/Amazon.prefab
- …
- Assets/Characters/Common/Stun.asset (common status effect shared by many characters)

Each folder could also hold the scriptableObjects for that character's unique powers and their status effects.

Common status effects (eg Stun, knockback) used by mor than one power could live in a Characters/Common/ folder

UI Prefabs and associated sprites belong in Assets/UI with subfolders such as Assets/UI/Quests

Localisation

Aspects that need to be factored into the development process from the beginning and not rushed at the end.

- Adjust in game assets as required for different regions
- Test game at all stages of development in all different versions
- Budget and plan for localization software, marketing, and ongoing content