
DRAGON RAGE
PRESS TO
START

TECHNICAL DESIGN DOCUMENT
Game Title: Dragon Rage
Author: Kadeem Cherman

Table Of Contents

Game Overview

Dragon Rage is a 3D survival game where players must endure waves of enemies while tactically using attacks, environment, and random drops to survive.
Levels feature different terrains, enemy types (dragons, ogres, high ogres, elemental dragons, monster beasts), and timers. Players can either:

- Survive until the timer ends
- Or defeat all enemies in the level

Randomized item drops grant temporary boosts like Health, Stamina, Block Shield (grace period), or an “Annihilation” power-up that clears the current wave. Enemies respawn after a set interval, keeping pressure high.

The gameplay loop emphasizes strategy + tactical combat rather than pure button-mashing.

Game Concepts

Player Control:

- WASD / left stick for movement
- Left click / RT for primary attack
- Right click / LT for block / shield
- “E” / Y to use special power-ups
- Sprint and dodge (shift / B button)

Combat & Enemies:

- Enemy waves spawn dynamically per level
- Enemy AI uses a finite state machine (Idle, Chase, Attack, Cooldown, Death)
- Difficulty ramps up each wave; stronger variants like high ogre or elemental dragon appear later

Collectibles / Power-ups:

- Health potion — restore HP
- Stamina potion — restore stamina for attacks/dodges
- Block Shield — brief invulnerability
- “Board Clear” Orb — destroys all current enemies; next wave resumes after respawn delay

UI:

- Health / Stamina bars
- Timer for level survival
- Wave count indicator
- Inventory of collected power-ups
- Mini-map / enemy markers

Physics Concepts

1. Kinematics & Rigidbody
    - Player uses a Character Controller with velocity-based movement

    - Enemies use Rigidbody NavMeshAgents for pathfinding
2. Collision & Triggers
    - Power-ups use trigger colliders (OnTriggerEnter)
    - Enemy attacks and player hitboxes use colliders to apply damage
    - Shield states toggle collider layers to grant invulnerability
3. NavMesh Pathfinding
    - Enemies navigate terrain using baked NavMesh and dynamic obstacles

---

Player Goals & Conditions

Objectives:

- Survive until timer reaches zero OR defeat all enemies in the level

Win Conditions:

- Timer expires while player still alive
- OR all enemies defeated before timer ends

Lose Conditions:

- Player HP reaches zero
- Optional: running out of lives or retries

Scoring (Optional):

- Bonus points for enemies defeated
- Extra points for time remaining

Tasks Breakdown & Time Estimates

Task #	Description	Est. Time
1	Setup Unity Project, folder structure, Git repo	30 min

Task #	Description	Est. Time
2	Create player prefab with movement, attack, dodge	2 hrs
3	Implement Player UI (HP, Stamina, Timer, Power-ups)	1.5 hrs
4	Create base enemy prefab + FSM (Idle, Chase, Attack, Death)	3 hrs
5	Implement NavMesh pathfinding for enemies	2 hrs
6	Create power-up prefabs (health, stamina, shield, board clear) with pickup logic	2 hrs
7	Wave / Level Manager for spawning enemies & timing	2 hrs
8	Add win/lose conditions, restart loop	1.5 hrs
9	Design Level 1 layout and enemy placements	2 hrs
10	Playtesting & bug fixing	3-5 hrs
11	Visual polish (lighting, VFX for power-ups, sounds)	2 hrs
12	Final build and submission	30 min

Total Estimated Time: ~22–24 Hours

Prototyping Features / Enemy FSM & Pathfinding

Enemy FSM States:

- Idle/Patrol: Waiting for player or roaming
- Chase: Move toward player (NavMesh)
- Attack: Melee or ranged based on enemy type
- Cooldown: Brief pause before next action
- Death: Play animation, disable AI, trigger respawn

Wave Manager:

- Controls spawn points and wave timers

- Randomly chooses enemy types per wave
- After “Board Clear” power-up used, all enemies destroyed and respawn timer starts

Level Progression:

- Each level introduces new terrain hazards and stronger enemies
- Timer or enemy count defines the win condition per level
- Boss fights at key milestones

Next Iterations:

- Implement multiplayer / co-op mode
- Add skill tree or ability upgrades between levels
- Add environmental traps (lava, collapsing bridges)
- Add save/load system
