Proj: 2D Tower Defense Game

Deadline: Thursday 26th June 2025, 11:59 pm

Game Overview

Title: "Crystal Defenders"
Genre: 2D Tower Defense *(Example1, Example2)*
Platform: PC (Godot Engine)
Team Size: 3 Students Max
Development Time: About 2 weeks

Core Concept

A simple tower defense game where players place towers to defend against waves of enemies. Focus on getting core mechanics working well with polished presentation.

Core Features (must implement)

Screens Required

1. **Main Menu** - Start, Quit, Credits
2. **Game Scene** - The actual tower defense gameplay
3. **Game Over Screen** - Victory/Defeat with stats

Maps

- **1 single well-designed map**
- Clear enemy path from spawn point to goal
- Designated tower spots (atleast 10-15 potential spots)

Towers (2 Types Only)

1. **Arrow Tower** - Single target, medium range, medium damage
2. **Cannon Tower** - Area damage, short range, slow fire rate

Enemies (3 Types Only)

1. **Basic Enemy** - Standard stats
2. **Fast Enemy** - Low health, high speed
3. **Tank Enemy** - High health, slow speed

Game Systems

- **10 waves** total (increasing difficulty)
- **Simple currency** (gold from kills)
- **Lives system** (10 lives)
- **No upgrades** - towers work at single level

Individual Responsibilities

Student A: Tower System

Task	Description	Deliverables
Tower Placement	Click-to-place with validation	`TowerPlacement.gd`
Arrow Tower	Single target, medium range	`ArrowTower.gd`
Cannon Tower	AOE damage, short range	`CannonTower.gd`
Projectiles	Movement, collision, effects	`ProjectileManager.gd`
Visual Polish	Range indicators, particles	Tower effects
Tower Selling	50% gold refund	Sell functionality

Branch: `feature/tower-system`
Minimum Commits: 10 commits

Student B: Enemy System

Task	Description	Deliverables
Path Movement	Enemies follow predetermined path	`PathFollower.gd`
Enemy Types	Basic, Fast, Tank variants	`BasicEnemy.gd` , etc.
Wave Manager	Spawn timing and quantities	`WaveManager.gd`
Health System	Health bars above enemies	Health display
Death Effects	Particles, gold reward	`EnemyDeath.gd`
Visual Polish	Hit effects, spawn particles	Enemy animations

Branch: `feature/enemy-system`

Minimum Commits: 10 commits

Student C: UI & Game Management

Task	Description	Deliverables
Main Menu	Start screen with options	`MainMenu.tscn/.gd`
Game Core	State management, pause	`GameManager.gd`
HUD	Gold, lives, wave display	`HUD.tscn/.gd`
Economy	Gold tracking and spending	`ResourceManager.gd`
Game Over	Victory/defeat with stats	`GameOver.tscn/.gd`
Audio	Background music, SFX	`AudioManager.gd`

Branch: `feature/game-ui`

Minimum Commits: 10 commits

Shared Responsibilities

- **Map Design & Visuals** - Design together, make it look good (3 hours)
- **Color Palette** - Agree on consistent colors (30 min)

- **Game Feel** - Particle effects, screen shake, eye candy (2 hours)
- **Testing & Balancing** - Everyone tests everything (2 hours)

Visual Polish Guidelines

Consistent Art Style

- Choose either pixel art OR vector style (not both)
- Use consistent color palette (5-6 colors max)
- Enemy and tower sizes should feel right

Recommended Polish Elements

1. **Particles** - Enemy death, tower shooting
2. **UI Feedback** - Button hovers, gold pickup animation
3. **Health Bars** - Clean, readable, consistent size
4. **Path Visualization** - Clear but not intrusive
5. **Background** - Simple but not empty

Recommended Free Assets

- **Art**: Kenney.nl (consistent style)
- **Fonts**: Google Fonts (pick 1 for headers, 1 for body)
- **Sound**: Freesound.org, Zapsplat
- **Music**: OpenGameArt.org

UI/UX Flow

```
Main Menu
    ├── Start → Game Screen
    ├── Credits → Credits Popup → Back
    └── Quit → Exit Game

Game Screen
    ├── Pause → Pause Menu
    │           ├── Resume
    │           └── Main Menu
    └── Game End → Game Over Screen
                    ├── Main Menu
                    └── Quit
```

Minimum Polish Requirements

Main Menu

- [ ] Background image/pattern
- [ ] Title with nice font
- [ ] Buttons with hover states
- [ ] Simple fade transition to game

Game Screen

- [ ] Clean, readable HUD
- [ ] Visual feedback for actions (place tower, kill enemy)
- [ ] At least 3 sound effects (place, shoot, enemy death)
- [ ] Background music (can loop)

Game Over Screen

- [ ] Clear victory/defeat message
- [ ] Show statistics (waves survived, enemies killed)
- [ ] Smooth transition from gameplay
- [ ] Options to replay or return to menu

Common Polish Pitfalls to Avoid

- ❌ Inconsistent art (mixing pixel art with vectors)
- ❌ No visual feedback for interactions
- ❌ Jarring scene transitions
- ❌ Unreadable UI (bad contrast, tiny text)
- ❌ Forgetting sound completely
- ❌ Placeholder graphics in final build

Quick Polish Wins

- ✅ Use tweens for smooth movement
- ✅ Add simple particles (Godot's CPUParticles2D)
- ✅ Screen shake on enemy death (subtle!)
- ✅ Consistent button style across all screens
- ✅ Use AnimationPlayer for UI animations
- ✅ Simple color variations for enemy types

Example Code Structure

Scene Hierarchy

Following is a suggested scene hierarchy for the game. You are free to modify it as needed, but this should give you a good starting point:

```
Main
├── MainMenu
├── GameScene
│   ├── GameManager
│   ├── UILayer
│   │   ├── HUD
│   │   └── PauseMenu
│   ├── GameWorld
│   │   ├── Map
│   │   ├── TowerContainer
│   │   └── EnemyContainer
│   └── AudioManager
└── GameOverScreen
```

Signal Flow for Polish

The following code snippets illustrate how to implement some of the polish features mentioned in the project requirements:

```
# Enemy killed - trigger multiple effects
signal enemy_killed(position, gold_value)

# Connect in GameManager
func _on_enemy_killed(pos, gold):
    spawn_death_particles(pos)
    spawn_gold_popup(pos, gold)
    play_death_sound()
    add_gold(gold)
    camera_shake(0.1)
```

With these additions, the game should feel complete and polished despite the short timeline. Remember: **simple but polished > complex but rough!**

Game Balance Reference

Towers

Type	Cost	Damage	Range	Fire Rate
Arrow	50g	10	150	1.0/sec
Cannon	100g	30	100	0.5/sec

Enemies

Type	Health	Speed	Gold Reward
Basic	50	50	10g
Fast	30	100	15g
Tank	200	25	30g

Starting Resources: 200 gold, 10 lives

Git Requirements

Commit Guidelines

- **Minimum:** 10 commits per student
- **Distribution:** Spread across at least 7 days **(SUPER important!)**
- **Size:** 50-300 lines per commit (avoid huge commits)
- **Messages:** Clear and descriptive

Marking Scheme (100%)

Component	Weight	Criteria
Individual Work	**60%**	
Core Functionality	20%	Features work as specified, bug-free
Git Commits	20%	Min 10 commits, spread over at least 7 days, meaningful messages
Code Quality	10%	Clean, commented, organized code
Individual Polish	10%	Visual effects, optimization in assigned area
Team Work	**40%**	
Integration	10%	Systems work together smoothly
Visual Polish	10%	Consistent art, particles, game feel
Game Balance	10%	Fun, appropriate difficulty
Completeness	10%	All screens work, no placeholders

Submission Requirements

1. Git repository with full history
2. README with game play instructions *(if different from default)*
3. Individual reflection *(each student, 1 paragraph each)*

Important Notes

- **Quality > Quantity:** Polish 2 towers rather than rushing 4
- **Test Early:** Don't wait until day 7 to integrate
- **Commit Often:** Show your work progress
- **Communicate:** Ask for help when stuck