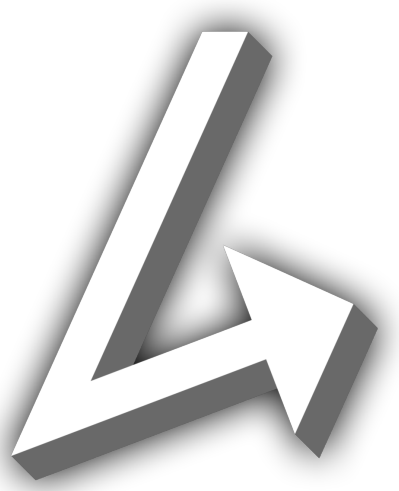
EZ RPG (MVP)

PolyDart™ x BinaryByte Production



Revision: 1.0.0

Concept Art (AI) **4**
Overview **5**
- Theme / Setting / Genre 5
- Core Gameplay Mechanics Brief 5
- Platform & Monetization 5
- Project Scope 5
- Influences (Brief) 5

The Pitch **6**
Description 6
What sets this project apart? 6
Mechanics **7**
- Player 7
- Sword 9
- Enemy 9
- Projectile 11
- Enemy Spawner 11
- ItemDrop 12
- Item 12
- ShopNPC 12
- ShopMenu 13
- Save/Load 13
- Player Equipment Visual 13
- Info Sidebar 13
- Info BottomBar 14
- MainMenu 14
- PauseMenu 14
- Settings 14
- SettingsMenu 14
- KeybindsMenu 15

“Things” **16**
- Player 16
- Environment 16
- Merchant NPC 17
- Food 17
- Wood Sword 17
- Stone Sword 17
- Iron Sword 17
- Diamond Sword 17
- Coin 18

- Jelly....18
- Wolf Pelt....18
- Bones....18
- Slime....18
- Wolf....18
- Skeleton....19
- Dragon....19
- Dragon Fireball....19

Schedule....20

- Mechanics (Isolated)....20
- Mechanics (Together)....20
- Things, Imports, and Assets....20
- Playtest, Polish, and Publish....20

Concept Art (AI)

Use these for inspiration. These are not meant to be directly what it should look like at the end,

at all.

Overview

- **Theme / Setting / Genre**
    - Medieval, Fantasy,
    - LowPoly, SmoothPoly, LowDetail, LowDefinitionTexturing,
    - Vivid, Colorful,

- **Core Gameplay Mechanics Brief**
    - Semi Openworld
    - WASD to movement
    - Shift to sprint
        - No sprint cooldown for MVP
    - Right click to interact
    - Hold right click to eat
    - Left click attack
    - Hold left click to auto-attack
    - Enemies damage players by getting too close

- **Platform & Monetization**

  PC, Buyable Game

- **Project Scope**
    - 2 Months to MVP
        - 2 Week sprints
    - 2 Roles
        - Code (Developers / Engineers)
            - Implementer of “if xyz does this, then abc does that”
        - Art (Assets / Makers)
            - Getter of “we need xyz”

- **Influences (Brief)**
    - TLoZ:WW, TLoZ:BotW, Elden Ring, Roblox, Minecraft

The Pitch

A small 3D first-person RPG game that's minimalistic and accessible, but engages players in an enjoyable chill adventure. This should be a good template for anyone pursuing RPGs or "3D first person game with basic combat and interactions"

House → Slime Plains → Forest → Mountain Base → Peak (Dragon) → Victory

Description

The player spawns in the house, interacts with a chest, and receives a wooden sword, unlocking the attack ability. Exiting the house, the player encounters Slimes wandering the plains outside the castle walls. Slimes can be defeated by repeatedly attacking them with the sword until their health is depleted. Upon a slime's death, it drops jelly, which the player can collect.

The player returns to the village and sells the jelly to the Merchant, then purchases a sword upgrade. They venture further out, past the Slimes, into a forest inhabited by Wolves. Combat is similar, and defeating Wolves yields wolf pelts. The player returns to the village, sells the wolf pelts to the Merchant, and buys another sword upgrade.

Next, the player moves beyond the Wolves to the base of a mountain, where Skeletons wander near the entrance to a stone staircase. Combat remains similar, and defeating Skeletons provides skeleton bones. The player returns to the village, sells the bones to the Merchant, and acquires a final sword upgrade.

Finally, the player ascends the mountain to confront a Dragon at the summit. This is a boss combat encounter where the Dragon shoots fireballs and periodically hovers in the air to dodge attacks. Successfully defeating the Dragon concludes the game.

What sets this project apart?

- Accessibility: low amount of controls
- Lore Style: minimalistic nature leads to "pause it and continue again whenever"
    - e.g. the first sword is wooden, and a dragon is just a "dragon"
    - No worries about forgetting important details after pausing
- Stability: Lower scope = faster and more stable game
  Freedom: Open world + simple = less force needed to push player in one direction

Mechanics

Think of these as the Base Prefabs and their Components
These **should** detail any abstractions or generic functionalities.
These **should not** detail game-definition specifics, these are all abstract.
These **should not** detail any Visuals, Audio, or Display resources (sprites, icons, etc).

- **Player**
  - Stats
    - Position
    - Rotation
    - CurrentSword
    - CurrentHealth
      - *For MVP, you could hard-code the MaxHealth*
    - Item(s)
    - Coins
    - Foods
      - *For MVP, this could just be an integer amount of Food that the character has left to hold right click and heal from.*
      - *For MVP, you could hard-code healAmount*
  - States
    - IsStanding, IsWalking, IsGrounded, IsRunning, IsJumping, etc (movement states)
      - *These are more open-ended since this will be first-person; we don't really need them for animating. But for sounds we'll want them - sounds could also be done directly when actions happen though. Whichever is fine!*
      - *Some could also just be lambdas of others for more readability when coding.*
        e.g. IsStanding => !IsWalking && !IsRunning && IsGrounded
    - ~~IsFighting (when in med-range of enemies)~~
      - Omitted from MVP in Week 5.
    - ~~IsNearDeath (when CurrentHealth <= MaxHealth / 10)~~
      - Omitted from MVP in Week 5.
    - IsDead (CurrentHealth <= 0)
  - Jump()
    - Move up, in the Y direction
  - Move()
    - Move in the X and/or Z direction depending on where we're looking
    - Sprinting
      - Activated by pressing Shift

            - No limit or cooldown for the MVP
    - Teleport()
        - Should be pretty self explanatory, but this moves a Player to a certain position
    - Respawn()
        - Teleport() all the way back to where we started
            - *The game startpoint could be wherever the player is OnAwake? Each load of the game?*
        - Reset CurrentHealth to max
    - Interact()
        - Called via a Right Click
        - Raycast from center of screen
            - *Distance for raycast should be hard-coded?*
        - If it's a shopNPC, then interact
            - *During MVP it may be cool to make an IInteract interface or something, but it's not necessary.*
    - Attack()
        - Called via a Left Click
        - Raycast from center of screen
            - Distance for raycast should be something like Sword.Stats.ReachAmount
        - Determine damage based on time since last Attack call and Sword's damage amount
            - Call Enemy's TakeDamage for that calculated damage
            - For this, maybe have a "MaximumDamageCooldown"?
            - *This still lets players spam the attack, for fun or to quickly make small amounts of damage*
    - AutoAttack()
        - Called while holding left-click
        - If we can do maximum damage (i.e. not on cooldown) then swing, otherwise bail and don't do so.
    - Eat()
        - Called while holding right-click
        - If not on cooldown
            - Heal()
                - *For MVP, HealAmount from eating can be hard-coded.*
    - TakeDamage()
        - Subtract from Health
            - Die() if Health was less than maximum
    - OnCollision()
        - With Enemy
            - Take Enemy.Stats.Damage
            - *Could be a spherecast or overlap*

        - With Projectile
            - Take Projectile.DamageAmount
            - Kill Projectile
        - With ItemDrop
            - Get ItemDrop.Item
            - Kill ItemDrop
    - Die()
        - Respawn()
            - *For the MVP this shouldn't have much punishment*

- **Sword**
    - Stats
        - Price
        - DamageAmount
        - ReachAmount
            - *Could hard-code this in MVP*
        - AttackCooldownDuration
            - *Could hard-code this in MVP*

- **Enemy**
    - Stats
        - Health
        - MaxHealth
        - WalkSpeed
        - WanderSpeed
        - ItemDrop(s)
        - Projectile(s)
        - CanShootProjectiles
            - If no Projectile(s), then clearly this wouldn't fire projectiles, so you may just want to lambda to Projectiles.Count <= 0
        - ShootCooldownDuration
        - HomePoint
            - Most likely placed where they spawned
            - [See Wander()]
        - FightingRadius
        - CanLeap
        - LeapLength
        - LeapSpeed
        - LeapHeight
        - CanHover
        - HoverDuration

        - HoverCooldownDuration
        - HoverDistance
    - States
        - AttackCooldown
        - HoverCooldown
        - IsWandering
        - IsFighting (when in FightingRadius of enemies)
        - IsNearDeath (when lower than of CurrentHealth)
        - IsDead (probably a lambda to CurrentHealth <= 0)
        - IsHovering
        - IsGrounded
    - Move()
        - Go at a direction of X and/or Z depending on where we're looking
    - Update()
        - [HandleStates]
        - if IsWandering
            - Wander()
        - If IsFighting
            - Walk()
                - Towards player
        - if IsFighting and CanHover
            - Hover()
        - if IsFighting and not IsHovering and not on cooldown
            - Leap()
        - if IsFighting
            - Walk()
    - Wander()
        - Pick a random spot near the HomePoint and walk to it slowly
    - Leap()
        - Move towards player at LeapLength, LeapSpeed, and move up LeapHeight
            - *No one should expect Enemy leaps to be simple, I can't properly plan how this should work. If we have to bail this during the development of MVP, then I don't see that as a problem.*
    - Hover()
        - If not on cooldown and not jumping
            - Transition above the ground at a Stats.HoverDistance
            - Hold the position for the HoverDuration
            - *Depending on how this ends up being implemented, along with the "reach" of player attacking, maybe make the Enemy invincible when hovering? This could be a bit of scope-creep, but just an idea. I think it would maybe be a cooler solution if enemies aren't invincible, but perhaps the player must jump to hit them when they hover? This will be a tricky thing to figure out, but should be fun though.*

- Look()
  - If on attack cooldown
    - then don't do anything
  - If IsFighting
    - LookAt, towards player
  - If IsWandering
    - LookAt movement direction
- TakeDamage()
  - If CurrentHealth <= 0
    - Die()
- Shoot()
  - Pick a projectile and launch it from our position
- Die()
  - spawn ItemDrop(s) at location
    - *The way I'd assume this could be coded is that at least one item will drop, and then the rest roll a 50% chance.*

- **Projectile**
  - Stats
    - Speed
    - Radius
    - DamageAmount
    - Target transform/position
  - Update()
    - LookTowards target
    - MoveTowards target
    - Explode once reached target
  - Explode()
    - Damage player within it for DamageAmount

- **Enemy Spawner**
  - Stats
    - Enemy
    - Radius
    - Shape
    - Spawnrate
    - SpawnTimer
    - CurrentSpawnsCount
    - MaxSpawns
    - Ground LayerMask

                - This is for the raycast down to find where to place a spawned enemy (so literally most likely the ‘ground’ layer)
                    - e.g. we want wolves spawning on the ground, not on trees.
    - Update()
        - SpawnTimer -= Time.deltaTime
        - If SpawnTimer <= 0 && CurrentSpawnsCount <= MaxSpawns
            - SpawnEnemy
            - Reset SpawnTimer to Spawnrate
    - SpawnEnemy()
        - Find a spot within Shape size of Radius centered at transform
            - Raycast down from arbitrary height *(test to find a good one)*
                - instantiate Enemy
                - Increment CurrentSpawnsCount
            - *If raycast fails to find a good spawn location, try spawning again at a new spot?*

- **ItemDrop**
    - Stats
        - Item
        - Position
        - PickupRange / Collider size

- **Item**
    - Stats
        - *The MVP won’t have an inventory management system so this is why Items shouldn’t be crazy.*
        - Icon?
        - Name?
        - Value
        -

- **ShopNPC**
    - Stats
        - Sword(s)
        - Food(s)
    - Interact()
        - Toggle ShopMenu’s visibility to true

- **ShopMenu**
    - Exit button
        - turns ShopMenu invisible onclick
    - Swords
        - Icon
        - Name
        - Price
    - Food
        - Food Icon
        - Food Price
    - BuySword()
        - If Player can afford it && their sword is less value
            - set Player's current sword to it
    - BuyFood()
        - If Player can afford it
            - increment Player's food

- **Save/Load**
    - *The objects saved/loaded should be implemented in the "glue things together" phase of development.*
    - Auto-save periodically
        - but expose a manual save method for PauseMenu
    - Save()/Load()
        - Player
            - Stats

- **Player Equipment Visual**
    - *Name can be changed, i didn't know what to call it*
    - ToggleVisual()
        - *(or SetVisibility() or SetVisible() etc, you do you)*
    - Separate Camera rendering with just a sword/equipment view.
        - should have two transforms that would be the left and right "hands" for the player
            - *For MVP, you only need to implement a right hand*

- **Info Sidebar**
    - from Player Stats
        - Coins Icon + amount
        - Jelly Icon + amount

        - Wolf Pelt Icon + amount
        - Bones Icon + amount

- **Info BottomBar**
    - from Player Stats
        - Health bar *(or heart amount display, whichever is decided)*
        - ~~CurrentSword icon + 'level'~~
            - *(I'm not sure if level should be shown, maybe discuss this)*
                - *NOT IN THE MVP*
        - Food icon + Foods amount

- **MainMenu**
    - **Highly suggest letting Lucas do this.**
    - Play button
    - Settings button
    - Quit button

- **PauseMenu**
    - **Highly suggest letting Lucas do this. (PolyDart Core has this handled)**
    - (Does not pause time in game.)
    - Unpause button
    - Save button
    - Quit button
        - Save()

- **Settings**
    - **Highly suggest letting Lucas do this. (PolyDart Core has this handled)**
    - SoundVolume Volume
    - MusicVolume Volume
    - Sensitivity
    - Invert
    - ScreenMode
    - VSynce
    - Keybinds

- **SettingsMenu**
    - **Highly suggest letting Lucas do this. (PolyDart Core has this handled)**
    - go-to KeybindsMenu button

    - SoundVolume slider
    - MusicVolume slider
    - Sensitivity slider
    - Invert toggle
    - ScreenMode toggle
        - Fullscreen, windowed, etc.
    - VSync toggle

- **KeybindsMenu**
    - **Highly suggest letting Lucas do this. (PolyDart Core has this handled)**
    - Interact / Hold to Eat
        - Right click
    - Attack
        - Left click
    - Jump
        - Space
    - Sprint
        - Shift
    - Pause / Unpause
        - Esc
    - *No plans to make WASD/Movement rebindable*

“Things”

This is where we define the “things” of the game.
These **should not** detail any abstractions or generic functionalities.
Think of these as the variants or game-specific instances for core Prefabs.
These **should** detail Mechanics, Visuals, Audio, or Display (sprites, icons, etc) resources and more if needed for the certain defined “Thing”.

- **Player**
    - Mechanics
        - Player
    - Health
        - Display
            - Icon
    - Attack
        - Visual
            - Sword swinging animation?
    - Eat
        - Audio

- **Environment**
    - Village
        - Player House Exterior
            - Visuals
                - Model
        - Player House Interior
            - Visuals
                - Model
    - Plains
        - SlimeSpawner
            - Mechanics
                - EnemySpawner
        - Ground
            - Visuals
                - GroundTexture
    - Forest
        - WolfSpawner
            - Mechanics
                - EnemySpawner
        - Ground
            - Visuals

                - Models
        - Mountain
            - SkeletonSpawner
                - Mechanics
                    - EnemySpawner
            - Ground
                - Visuals
                    - Texture
            - Staircase
                - Visuals
                    - Models
- **Merchant NPC**
    - Mechanics
        - ShopNPC
    - Visuals
        - Model
- **Food**
    - Display
        - Icon
- **Wood Sword**
    - Mechanics
        - Sword
    - Visuals
        - Model
    - Display
        - Icon
- **Stone Sword**
    - Mechanics
        - Sword
    - Visuals
        - Model
    - Display
        - Icon
- **Iron Sword**
    - Mechanics
        - Sword
    - Visuals
        - Model
    - Display
        - Icon
- **Diamond Sword**
    - Mechanics

        - Sword
    - Visuals
        - Model
    - Display
        - Icon
- **Coin**
    - Display
        - Icon
- **Jelly**
    - Mechanics
        - ItemDrop
    - Visuals
        - Model
    - Display
        - Icon
- **Wolf Pelt**
    - Mechanics
        - ItemDrop
    - Visuals
        - Model
    - Display
        - Icon
- **Bones**
    - Mechanics
        - ItemDrop
    - Visuals
        - Model
    - Display
        - Model
- **Slime**
    - Mechanics
        - Enemy
    - Visuals
        - Model
    - Audio
        - Death
- **Wolf**
    - Mechanics
        - Enemy
    - Visuals
        - Model
    - Audio

        - Death
- **Skeleton**
    - Mechanics
        - Enemy
    - Visuals
        - Model
    - Audio
        - Death
- **Dragon**
    - Mechanics
        - Enemy
    - Visuals
        - Model
    - Audio
        - ShootFireball
        - Hover
        - Death

- **Dragon Fireball**
    - Mechanics
        - Projectile
    - Visuals
        - Model/VFX
    - Audio
        - Explode

Schedule

- **Mechanics (Isolated)**
    - Week 1 + 2
        - No art. No glue-ing together tasks. Isolated testing.
        - Focus on coding the Mechanics that can be isolated.
        - Solve in ways that you think others should be able to use with other things, or that are easily modifiable to be used with other things.
            - But most importantly, keep your stuff isolated. If it doesn't need to depend on other mechanics, then don't make it depend on other mechanics. e.g. an Enemy should be testable in this phase via OnGUI buttons, and not require the player mechanics.

- **Mechanics (Together)**
    - Week 3
        - **Important:** At this stage, we should have all isolated Mechanics "done" per their rough descriptions.
        - This is where things get glued together.
        - It's expected that some mechanics will be changed during this week.
    - Week 4
        - After things are glued together, this is where a game-loop forms.
        - This is the final week where you should tweak mechanics.
        - Acting in preparation for the next week is key during this week.

- **Things, Imports, and Assets**
    - Week 5 + 6
        - Time to go through all of the "Things" and define them.
        - This is also when we add assets to the project, or make art, sounds… etc.

- **Playtest, Polish, and Publish**
    - Week 7
        - MAJOR CODE FREEZE! *Unless we absolutely must deploy hotfixes to bugs.*
        - This is where you gather testers, and receive feedback.
    - Week 8
        - Spend this week prepping ads, capsule art, storefronts, etc, and preparing the finalized game for the masses.
        - It is not expected that the game will fully publish on anticipated platforms during this week. Game publishing is usually dependent on the publishing platform itself.