D.I.G.R. Discovery Intergalactic Resources

(Patrick Brister, Hyrum Hovey, Michael Rios)

Summary **3**

Gameplay (Core Loop) **4**

Gameplay (Contextual Notes) **4**

Multiplayer Support **5**
- Multiplayer Support will be added in v1.2 5

Art Style **6**
- UI/Menu System 6
- Environment 6
- Effects 6
- Player Buildings & Units 6
- Alien Buildings & Units 7
- Art Prioritization 7

UI/Control System **8**
- UI Engine 8
- Controls (Menus) 8
- Player Controls (Game) 8

Units **11**
- Player Unit 11
- Players Units 13
- Alien Units 15

Artificial Intelligence Systems **17**
- System Basics 17
- Player AI Systems 17
- Alien AI Systems 18
- World Brain (World AI System) 18

Level System **19**
- Level Engine 19
  - Universal Level Rules 19
    - Winning Conditions: 19
    - Losing Conditions: 19
    - Other Conditions and Factors: 19
  - Level Engine Components 19

Terrain Object 19
Exterior Level Wall 19
Game Ceiling 20
Rock Pillars 20
Level Features 21
Hazards: 21
Red Gas: 21
Invincible Stone: 22
Resources 22
Fragments 22
Pulse 22
Notes on Resources: 22
Resource Spawner 22
Level Structures 23
Player Base 23
Lift 23
Static Walls 23
The Repair/Refit Pad 23
Movable Objects 23
Sensors 23
Traps 24
Collectors 24
Alien Structures 24
Spawning Structures 24
Defensive Structures 24
Winning Node 25
Standard Node 25
Final Node 25

Technical Specs 26
Hardware Target 26
Development Engine 26
Development Assets 26

Summary

- Game Summary:
    - The player is in charge of the a robotic mining operation being conducted on behalf of the super-corporation known as Discovery Intergalactic Resources (DIGR). DIGR is tasked with the mission of connecting new superficial planetary colonies to a power source deep in the rock layers of the planet.
    - Completing this missing and finding a stable power source for the colonies will ensure their survival and without it - they are doomed.

Gameplay (Core Loop)

- The core gameplay loop is as follows:
    - The player begins the level in their Mobile Command Unit (MCU) in first person view.
    - The player can begin drilling out rock.
    - The players units spawn. (This occurs every X seconds)
    - As the player drills away rock they gain resources.
    - Workers automatically take resources from the MCU to the Lift to secure them.
        - This worker loop between the MCU and the lift is automatic.
    - The player can deploy their MCU and enter command mode.
        - Command mode allows the player to give units commands.
        - Command mode allows the player to select what units will come down the lift.
        - Command mode gives the player an overhead view.
    - The player will attempt to find the entry point to the next level by drilling out rocks.
    - Aliens will occasionally attack the player and their workers.
    - The player will use command view to maintain defense of their workers.
    - The player will win when they reach the next level entrance (or the final node)
        - The player will lose if the MCU or the Lift is destroyed.

Gameplay (Contextual Notes)

- The MCU and the workers create a line of units going back and forth.
    - The line will always be the pathfinding systems shortest route from the MCU to the Lift.
    - The player must defend the worker line using their soldiers.
- The MCU is weak in the wrong circumstances.
    - It can be overwhelmed if it is too far from soldiers and is attacked.
    - It is also virtually defenseless in Command mode
- The more rock that is mined away the more aggressive the aliens get.
- The player will easily lose if they spread their units too thin, fail to protect their worker line, or try to power through the level alone.

Multiplayer Support

Multiplayer Support will be added in v1.2

(v1.0 is final release*, v1.1 is first patch, v1.2 is addition of Multiplayer Support
*for the scope of this course this is the only planned iteration)

- Multiplayer mode will consist of a larger level where each player controls a robotic force complete with their own base and lift. Players will not fight one another, but will race to find the Winning Node.

Art Style

UI/Menu System

- The UI and Menus will have a sleek and clean futuristic look.
- Key design components:
    - Simple and Clean
    - Lots of straight lines
    - Selected areas glow white, unselected, ambient gray
    - UI fades slightly when there is no interaction

Environment

- Theme:
    - The environment will be a low poly theme that will use minimal processing power to allow for a more dynamic appearance.
- Minerals are embedded in rock
- Different depths have different type of rock
    - Beginning levels are brown and rocky
    - Medium levels are gray
    - End levels are rocky and molten

Effects

- Lighting and Effect assets will be used to allow for a modern appearance within the low poly look.
- Higher power systems will allow for heavy use of lighting.
    - Lighting effects will be used for all player units
        - The main player MCU unit will house several lights to allow for an immersive first person experience
    - Volumetric lighting will be implemented using the Aura asset if time and testing allows

Player Buildings & Units

- The units of DIGR are modern and sleek.
- All units are robots with illuminated eyepieces and light emitting diodes/details.
- Units are also low poly as to fit the rest of the world.

Alien Buildings & Units

- Aliens are organic and resemble some sort of fungus-bugs.
- Aliens have gross goo sacks.
- Aliens have stabbing and spitting abilities.
- Alien buildings pulsate.

Art Prioritization

- Primary focus for inhouse art development will be the Player's characters. This will be followed by the Alien AI. All other art assets will be brought in from the asset store.

- Art prioritization reference:

- Player Objects
    - Worker
    - Soldier
    - Sentries
    - MCU
    - Lift

- Aliens
    - Blips
    - Blorps
    - Blortons
    - Buildings

UI/Control System

UI Engine

- The UI has a clean and sleek look and feel.
    - As things are clicked on they respond by slightly changing tone and moving or changing size.
    - All UI actions should be confirmed with a visual and audio response
- The UI is grouped into sections so that it can be easily moved and customized during game design.
    - This may allow for different UI modes that could be easily implemented
    - The customization of the UI by the player can be added as a later feature.

Controls (Menus)

- This section is about control through the games menus.
    - Title Screen
    - Main Menu
        - New Game
            - Starts a new game
        - Load Game
            - Loads a saved game
        - Options
            - Music Volume
            - SFX Volume
        - Exit

Player Controls (Game)

- The overhead and first person UI’s are different
    - Overhead UI
        - Has Cursor
            - Script Required: CursorCrossActivator
                - HideCursor()
                    - Hides the cursor
                - ShowCursor()
                    - Shows the cursor

        - Allows for selection of units
            - Script Required: UnitSelectionManager
                - Click a unit to select

            - SelectUnit()
                - Selects a unit into the manager on click
        - Double click to select all units of that type on screen
            - SelectUnitsOfType_Visible()
                - Selects all of the units of the type clicked on click that are visible within the camera
        - Triple click to select all units of that type on level
            - SelectUnitsOfType()
                - Selects all of the units of the type clicked on click that are on the level
    - Note: There is no drag select for units to reduce programming overhead -> This may be added with a plugin in v1.1

- Shows information on Resources at all times
    - Script Required: ResourceInfoManager
        - GetResourceCounts()
            - This checks the information on all player resource stores and saves these values to variables for easy access via the UI display system.
                - fragmentsUsable
                    - Represents fragments that have been successfully delivered to the lift
                - fragmentsWIP
                    - Represents fragments that works in progress and are either held by workers, or by the MCU.
                - pulseUsable
                    - Represents pulse that has been successfully delivered to the lift
                - pulseWIP
                    - Represents pulse that works in progress and are either held by workers, or by the MCU.

- Show information on Units when selected
    - Script Required: UnitInfoManager
        - GetSelection(selection)
            - This determines if one or multiple units of been selected - and then determines how to display this information. Individual unit info won’t be displayed.
        - GetUnitInfo()
            - If only one unit was selected, call the units individual stats and display them.

            - Shows information on Buildings when selected
                - Script Required: UnitInfoManager
                    - GetSelection(selection)
                        - This determines the type of building selected and arranges how the information will be displayed.
                    - GetUnitInfo()
                        - This calls the buildings information and displays it.

    - First Person UI
        - Has Crosshair
            - Script Required: CursorCrossActivator
                - HideCrosshair()
                    - Hides the crosshair
                - ShowCrosshair()
                    - Shows the crosshair
        - Shows resources
            - Script Required: ResourceInfoManager
                - GetResourceCounts()
                    - This is the same method called by the Overhead UI.
        - Shows health
            - Script Required: UnitInfoManager
                - GetUnitInfo()
                    - Calls getunitinfo for the player unit
        - Shows energy
            - Script Required: UnitInfoManager
                - GetUnitInfo()
                    - Calls getunitinfo for the player unit
        - Shows info on rock in crosshair
            - Script Required: CrosshairAI
                - GetObjectAtCrosshair()
                    - This checks what type of object is hit by the crosshair ray
                - GetUnitInfo()
                    - This gets info if the item hit is a unit
                - GetRockInfo()
                    - This gets info if the item hit is a rock

Units

Player Unit

- The player unit is called the MCU and it has three main functions.
    - **The MCU is the player.**
        - Script Required: PlayerController
            - InputCheck()
                - Checks for all inputs
        - Script Required: PlayerHealthManager
            - CheckPlayerCondition()
                - Checks all of the players stats and does any required calculations
                - All adjusters happen here

    - **The MCU can drill away rock, and temporarily stores resources, while workers carry resources from the MCU to the main base.**
        - Script Required: PlayerHopperLoad
            - CheckHopperLoad()
                - Checks the amount of resources in the MCU hoppers
            - AlertWorkersOfHopperLoad()
                - Alerts workers if there is contents in the hopper when before there was not.
            - AlertWorkersOfEmptyHopper()
                - Alert workers if the hopper is empty and before it was not.

    - **The MCU can deploy and allow the player to command units.**
        - Script Required: PlayerController
            - SwitchToCommand()
                - Checks if the MCU can deploy, deploys the MCU, switches the view.
            - SwitchToFirstPerson()
                - Checks if the MCU can undeploy, switches to first person, undeploys the MCU.
            - DeployMCU()
                - Actually deploys the MCU with animations and the cutting/adjusting of controls.
            - UndeployMCU()
                - Undeploys the MCU with animations and restoring/adjusting of controls.

- The unit has a laser drill that helps mine away rock.
    - Script Required: DrillAI
        - Drill()
            - Calls the drilling coroutine
        - StopDrill()
            - Stops the drilling coroutine
        - Overheat()
            - When max heat hits, forces StopDrill to run
        - Cooldown()
            - When StopDrill runs and OverHeat has run, then starts a timer before the drill can restart
        - IncreaseHeat()
            - Increases heat to the heat value
        - DecreaseHeat()
            - Decreases heat from the heat value
        - Jam()
            - Similar to overheat but does not require max heat to be reached

- The unit also has a weapon to fight aliens and a method to defend itself.
    - Script Required: WeaponAI
        - Attack()
            - Calls attack coroutine loop
        - Block()
            - Calls block coroutine loop

- The unit will have one of two locomotions:
    - A track or wheel of some kind
        - Benefit to this approach is ease of design and programming
    - Legs

Players Units

- The player units in the game will have AI and will function based on player commands. The player will not directly control these unit classes.
    - Workers
        - Workers can do only two things:
            - Move to MCU
                - FindMCU()
                    - Locates the MCU and sets up pathing
                - GotoMCU()
                    - Follows the path
            - Collect resources from MCU
                - CollectResourceFromMCU()
                    - Subtractrs resource from the MCU and adds it to the worker
            - Move To Lift
                - FindLift()
                    - Pathfinds to the lift.
                - GotoLift()
                    - Follows the path.
            - Deposit resources to Lift
                - DepositResourcesToLift()
                    - Subtracts the resource from the MCU and adds it to the usable resources.
    - Soldiers
        - Soldiers can do five things:
            - Move to areas based on commands from the overview view
                - MakePath()
                    - Make a path to the selected spot
                - Goto()
                    - Follow the path
            - Move to the MCU
                - MakePath(MCU)
                    - Makes a path to the MCU
                - GotoMCU()
                    - Follows the path
            - Move to the lift
                - MakePath(Lift)
                    - Makes a path to the lift
                - GotoLift()
                    - Follows the path
            - Patrol

            - Patrol()
                - Alternate between GotoLift and GotoMCU to patrol worker path
        - Range attack enemies on sight
            - CheckForEnemies()
                - A scan that checks for enemies nearby
            - TargetEnemy()
                - The selection of an enemy unit
            - AimToEnemy()
                - The aiming towards the enemy unit
            - AttackEnemy()
                - The actual weapon coroutine calls
- Sentries
    - Sentries can do six things:
        - Move to areas based on commands from the overview view
            - MakePath()
                - Sets a path to the selection area
            - Goto()
                - Follows the path
        - Move to the MCU
            - MakePath(MCU)
                - Makes a path to the MCU
            - GotoMCU()
                - Follows the path
        - Move to the lift
            - MakePath(Lift)
                - Makes a path to the lift
            - GotoLift()
                - Follows the path
        - Deploy
            - Deploy()
                - Checks if the area is deployable
                - Cut all movement and deploy
                - Run deploy animation
        - Range attack enemies on sight when deployed
            - CheckForEnemies()
                - A scan that checks for enemies nearby
            - TargetEnemy()
                - The selection of an enemy unit
            - AimToEnemy()
                - The aiming towards the enemy unit
            - AttackEnemy()
                - The actual weapon coroutine calls

        - Undeploy
            - Undeploy()
                - Checks that undeployment is currently possible
                - Restores control
                - Plays undeploy animations

Alien Units

- Alien units are simple in their abilities. There are three enemy units.
    - Blips
        - Blips are small creatures that run around the level and attack player units.
            - Roam()
                - AI for constant roaming and scanning for enemies
            - CheckForEnemies()
                - The actual scans for enemies nearby
            - TargetEnemy()
                - When an enemy is found - this handles targeting them
            - GotoEnemy()
                - When an enemy is targeted - this handles moving towards them
        - Blips will deal damage with melee attacks.
            - Attack()
                - The attack coroutine call goes in here
        - Blips will occasionally explode.
            - CheckTargetProximity()
                - Check if the target is within explode range
            - Explode()
                - Activate explosion and then destroy unit
    - Blorps
        - Blops are medium sized creatures that roam and attack.
            - Roam()
                - The standard roaming functionality.
            - CheckForEnemies()
                - The standard scan functionality.
            - TargetEnemy()
                - The standard target functionality.
            - GotoEnemy()
                - The standard movement towards an enemy.
        - Blorps will attack with a ranged attack.
            - Attack()
                - Initiation of the ranged attack coroutines.
        - Blorps will occasionally attack with a melee attack.
            - ChargeEnemy()

            - Charges the targeted enemy
        - AttackMelee()
            - Runs the melee attack coroutine
- Blortons
    - Blortons roam like other enemies but they are larger
        - Roam()
            - The standard roaming functionality.
        - CheckForEnemies()
            - The standard scan functionality.
        - TargetEnemy()
            - The standard target functionality.
        - GotoEnemy()
            - The standard movement towards an enemy.
    - Blortons will attack with a close range strong attack.
        - ChargeEnemy()
            - Charges the targeted enemy
        - AttackMelee()
            - Runs the melee attack coroutine

Artificial Intelligence Systems

System Basics

- The AI systems will use the following systems:
    - Navmesh for navigation
        - Most navigation routines will follow this pattern:
            - Get a Goto Point
            - Check if that point is accessible via Navmesh
            - If it is not, report back to get a new Goto Point
            - If the point is accessible - go to the point
    - A single collider for all collisions
        - Units, structures and environment will all use the same approach
            - All objects within the object do not have colliders
            - The parent object is the only object with a collider
            - The parent object holds AI scripts
            - All children are managed via the parent scripts
                - Children scripts that are used are activated/deactivated or implemented by the parent object scripts
    - Animations using the Unity animation system
        - The Mecanim system in unity will be used for managing all animations that are used on objects.

Player AI Systems

- Buildings
    - Building AI is simple and will typically be constructed using the following scripts on the parent object:
        - Animator
            - Handles animations
        - StatusManager
            - Handles health, shields and specific point systems for the object
        - AbilityManager
            - Handles the implementation and calls of all of the objects abilities
- Units
    - Unit AI consists of multiple components:
        - MainController
            - Handles the managing of all calls for the object
        - NavigatorController
            - Handles all the location and navigation techniques to move around the level

        - ActionController
            - Handles all of the actions and abilities of the object.

Alien AI Systems

- Buildings
    - Alien buildings will have one or two scripts depending on the building:
        - HealthManager
            - Handles all of the health and stats of the object
        - SpawnManager (if the building is a spawner)
            - Handles all of the spawning and creation of new enemy units
- Units
    - Alien units use the same mechanics as the player units, with one major difference. The alien units are controlled from a AlienBrain script that helps manage the collective actions of the alien team.
        - AlienBrain (method signatures only - self explanatory)
            - SomeBlipsRoam()
            - AllBlipsRoam()
            - SomeBlipsStop()
            - AllBlipsStop()
            - OneBlorpAttack()
            - AllBlorpAttacks()
            - SomeBlorpsRoam()
            - OneBlortonAttack()
            - AllBlortonsAttack()
            - AllHoldPosition()
            - AllAttack()
            - AllDefend()

World Brain (World AI System)

- The WorldBrain script will help manage all wide scope aspects of the game - such as when to switch levels or pause or pause.
    - The world brain will serve as a catch all for anything requiring a full project scope.

Level System

Level Engine

The single player missions in the game allow for a certain flexibility that requires the level system to be simplified and not overly complex. A standard level is laid out using a set of rules that are determined by our “Level Engine”.

Universal Level Rules

- Winning Conditions:
  - The player must reach, and mine away the “Winning Node” of the level.
- Losing Conditions:
  - The players lift door is broken down and the lift becomes infested with aliens.
- Other Conditions and Factors:
  - The player must mine out blocked areas before they are able to traverse them.
  - The player can only deploy items on empty areas.
  - The player can only have X amount of Fragments worth of units on the field at one time.

Level Engine Components

- Terrain Object
  - The terrain object will have several preset shapes that can be selected or set randomly. A range adjuster maintains a healthy Y scale value to keep the terrain flat enough to navigate naturally and rough enough to provide an interesting look.
  - The terrain object per level will use the same shader as the majority of the rock within the level to allow for a uniform look.
  - Several shaders will be available to allow for different looking levels.
- Exterior Level Wall
  - The exterior wall will be constructed of an object with a similar appearance to the stones within the mine, except these stones will be unbreakable. It will be

important to provide the player with feedback that these stones will not be minable.

    - (NOTE: We do not want the stones to look different than the rest of the cavern, so UI feedback may be a good approach to communicate this to the player.)

- Game Ceiling
  - The game ceiling is a simple object. Terrain objects cannot be flipped over in unity but so a mesh will have to be used. The mesh will act like a flipper terrain that will appear as a ceiling in the cavern.
    - (NOTE: Overhead cameras will not render the ceiling so that the player can command from an overhead view.)
- Rock Pillars
  - Rock pillars are a core component of the game. They make up the majority of the fill and minable objects within the cavern.
  - Rock pillars are parent objects of several children “stones” that are arranged into a column. Each stone in the column belongs to a group within the parent, there are four groups per rock pillar.
  - The Rock Pillar maintains a health value, and the appearance of the pillar reflects the health.
    - At 100% health all of the stones will be present and the pillar will be completely full.
    - At %75 health ¼ of the pillar will break away.
    - At %50 health ½ of the pillar will break away.
    - At %25 health ¾ of the pillar will break away.
    - At %0 health the pillar will completely break away.
  - Rock pillars will reveal objects as they are mined away. These objects can be positive or negative. They can also be incorporated into the Rock Pillar or simply hidden by it.
    - Rock Pillar Incorporated Objects
    - NOTE: These objects will be children of the pillar
      - Resources
        - Resources can be embedded into the stones
        - When stones are mine away the resources are collected
        - When the stone is gone the resource is gone
      - Explosives
        - Explosives can be embedded into the stones
        - When stones are mined away the explosives have a risk of damaging the miner and/or nearby units
        - At any level of health there is a chance of an explosion if explosives are present in the rock

                - Explosions of this type will destroy the rock and potentially cause a collapse.
            - When the stone is gone, the explosive is gone.
        - Collapses
            - When a collapse occurs the rock is destroyed, and more stones are dumped from above, physics will be used if possible, otherwise this will be animated.
            - A collapse will kill or damage any units that are hit by it.
            - A collapse can occur on its own or be triggered by an explosion.
            - The collapse object will be generated when the pillar is created only if it registers as a pillar that needs to have a collapse object by the spawner. This will reduce memory use.
    - Rock Pillar Hidden Objects
        - NOTE: This objects will NOT be children of the pillar - they will spawn randomly from a separate spawner. They will be revealed when the pillar is destroyed.
        - Resources
            - One time resource pickups
        - Powerups
            - One time power up pickups
        - Explosives
            - One time explosive drops
                - Option to pick up and move undetonated explosives
        - Red Gas
            - Red Gas drop will generate Red Gas that does damage to the player units.

Level Features

- Hazards:
    - Explosions when mining are possible and can damage your units.
    - Collapses can occur and sometimes allow for mining of the collapsed areas.

- Red Gas:
    - Red Gas is a gas emitted from the aliens and their secretions which can be harmful to the player. Depending on concentrations Red Gas can do the following:

  - Damage the players units
  - Slow the players units
  - Enhance the aliens attributes

- Invincible Stone:
  - These stones cannot be damaged, removed or mined away.
  - Used to limit movement
  - Used to cause chokepoints

Resources

- There are two resources types.
  - Fragments
    - are minerals, gems, and precious stones of all varieties. They all provide points to allow for the delivery of units from the Lift.
  - Pulse
    - is the lifeblood of the planet. It has healing properties, which allows for regeneration of units as well as various bonuses across the board.

- Notes on Resources:
  - Resources are part of the areas that are mined out. These are dropped resources and they are consumable.
  - Resource veins can be exploited with deployables, and are not consumable like dropped resources.

- Resource Spawner
  - The Resource Spawner will handle all of the non-pillar generated resources.
  - Resource Drops
    - Resource drops are one time drops that are hidden by pillars.
  - Resource Veins
    - Resource veins are also hidden by pillars, however they are not one time pick ups, but instead require collectors to be built on them.

Level Structures

- Player Base
    - There are many items that the player starts with that they have no control over building or rebuilding if destroyed. These items are placed at level generation by the designer or by the AI depending.

    - Lift
        - The lift is the main structure that the player must defend and the main structure that the aliens will attack.
        - The lift also provides the player with units. The lift will provide the player with X fragments worth of units every Y seconds.
        - The player will be able to select which units and how many of each will come down the lift at a time.
    - Static Walls
        - Static walls will be put around parts of the player base that will help defend them from aliens.
        - These will be more prominent in earlier levels.

    - The Repair/Refit Pad
        - The Repair/Refit Pad will automatically heal and enhance units that are near it.
        - The Pad can be destroyed and if it is destroyed, it will be gone for the remainder of the level.

    - Movable Objects
        - Movable objects are objects that can be deployed by the player in clear areas.

        - Sensors
            - Sensors can be deployed and allow for the detection of enemies through the rock that blocks the view of them.
            - The sensors will show enemies from the overhead view.
            - The sensors will show enemies from the minimap radar.
            - Only X amount of sensors can be deployed at any one time.
            - Sensors use Fragments when created or moved.
            - Sensors use Pulse when operational.

        - Traps
            - Traps can be deployed and will harm enemies when they cross them.
            - Only X amount of traps can be deployed at any one time.
            - Traps use Fragments when created or moved.
            - Traps use Pulse when standing by.

        - Collectors
            - Collectors require Fragments to build
            - Collectors do not require any resources to maintain
            - Only X amount of collector can be deployed at any one time.

- Alien Structures
    - Spawning Structures
        - Red Gas Spawners
            - Red Gas spawners simply spawn a large collider that with it carrys a "gas" effect.
            - When player units or buildings are colliding with a Red Gas object they will lose bonuses and will not regenerate.
        - Blip Spawners
            - Blip spawners will continuously spawn Blips until a max amount of Blips are created.
            - Blips are intended to die often and so spawners replace them as they die. This maintains a constant amount of blips to harass the player.
        - Variety Spawners
            - Variety spawners will continuously spawn random enemies.
            - They work exactly like Blip spawners except they spawn everything else.

    - Defensive Structures
        - Projectile Turrets
            - Projectile turrets will turn and fire towards player units when they get close.
            - These units will fire until they are destroyed.
        - Proximity Turrets
            - Proximity turrets do damage to units within a range of them.
            - They damage an area and can hurt many units at one time if they are not destroyed.

- Winning Node
    - The winning node has the same function in each level, reaching it, and mining it, wins the level.
        - Standard Node
            - Winning for most single player levels are blockages to a downward tunnel. Often these blockages will house some kind of boss or a number of hostile alien units. These types of winning nodes are used to show that the player has passed this level - but is still continuing down further into the ground.
        - Final Node
            - At the final level the single player game the winning node will not be a continuation down, but will be a massive Pulse well that will no doubt save your people.

Technical Specs

Hardware Target

- The game is intended to be released on the PC for mid-range to low-end systems.
    - Secondary release on Mac (Unplanned)
        - Tertiary release on Android/iOs (Unplanned)

Development Engine

- Unity (Version undecided)

Development Assets

- Assets to be used from the asset store
    - All environment assets will be pulled from Free kits on the Asset Store - some options are listed:
        - Free Hand Painted Rocks N Boulders by Poly Ninja
        - Low-Poly Resource Rocks by False Wisp Studios
        - Low poly styled rocks by Daniel Robnik
        - LowPoly Environment Pack by Korveen
        - Polygon Desert Pack by Runemark Studio

    - All Audio except for any required voices will be from Free kits on the Asset Store - some options are listed.
        - UI Sound Effects Collection Pack 2: Buttons by Nectar Sonic Lab
        - Electric Sound Effects Library by Little Robot Sound Factory

    - Shaders and Effects that may be used from the Asset Store - some options are listed:
        - Post Processing Stack by Unity Technologies
        - Fire & Spell Effects by Digital Ruby
        - KY Magic Effects Free by Kakky
        - DL Fantasy RPG Effects by dreamlevel
        - Aura - Volumetric Lighting by Raphael Ernaelsten