Node Hacker 2036

# Overview

Node Hacker will be a procedurally generated multi level experience where the player is tasked with navigating through a level, retrieving data caches protected by puzzle mini games. The layout of the levels will be patrolled by enemies. The main idea is to created a challenging game that also pokes fun of the current trend of re-imagining sci-fi settings through the lens of someone interpreting what a sci-fi creator from the 80’s would imagine our own future to look like.

## Emotions

1. Humor – tutorial vods, mission briefings
2. Nostalgia – retrowave style game meant to invoke memories of what the 80’s thought the future would look like
3. ~~Agitation – completing procedurally generated levels with puzzles and enemies~~

## Game Elements

1. ~~Procedurally generated levels~~
2. Required collectibles required to progress to the next level
3. Puzzle mini games to retrieve required collectibles
4. Optional collectibles
5. The… player
6. ~~Several~~ enemy types with different abilities and behaviors
7. ~~Player will have different weapons to combat the enemies –~~ no ammo, all cool down stuff
8. ~~Mission briefings and tutorials will depict a stereotypical hacker / decker with tongue and cheek humor about the current state of retrofied-future-sci-fi~~
9. Art assets inspired by C64’s Neuromancer, William Gibson’s novels, and the Shadowrun universe will provide additional challenges when navigating the levels

# Element Details

## Procedurally Generated Levels

~~Each level will consist of additively loading level “blocks” from a bank of prebuilt scenes into a conceptual grid and connected through doors. The size of the level will scale as the player progresses. Based on the level size, the game manager will determine the number of required collectibles, as well as spawn probabilities for optional collectibles and enemies.~~

Each “block” will have their own navmesh for enemy agents. There will be zero to several spawn zones for collectibles and enemies which will be used by the game manager to spawn said elements based on the aforementioned quotas and probabilities.

## Required Collectibles

The player will have to collect file fragments in order to beat the level. Once the data fragments are collected, you will have the option of jacking out (while out of combat).

## Puzzle Minigames

In order to decrypt a data fragment, you’ll have to solve a visual representation of a decryption algorithm. These will include game logics built in my puzzler project but not implemented:

1. ~~Simon pattern match~~
2. ~~Memory card game~~
3. ~~Operation on a mobo~~
4. ~~Image shift~~
5. Roller Ball (Added)

## Optional Collectibles

The map will spawn optional collectibles to help the player:

1. ~~Health packs – not sure what the representation will be for this, but it will repair your firewall~~
2. ~~SIN’s – ID’s worth real money! That you… totally can’t use in this game~~
3. VHS Tapes (Added)

## Player

Player will move via teleportation with a cooldown based on distance traveled. Will have to investigate if I can utilize a ~~navmesh~~ for teleportation layer? Might be easier to restrict movement. Will also need walls thick enough that you can’t look through or darken screen if player looks through.

## Enemies

1. ~~Watcher – if sees player, sets alert and all enemies in x range will set their nav agents destination to the player until the alert ends. Once they are in close range, they might teleport.~~
2. Standard Ice – Standard enemy, will shoot at mid range
3. ~~Black Ice – AOE based damage radiates from enemy’s origin~~

## Weapons / Abilities

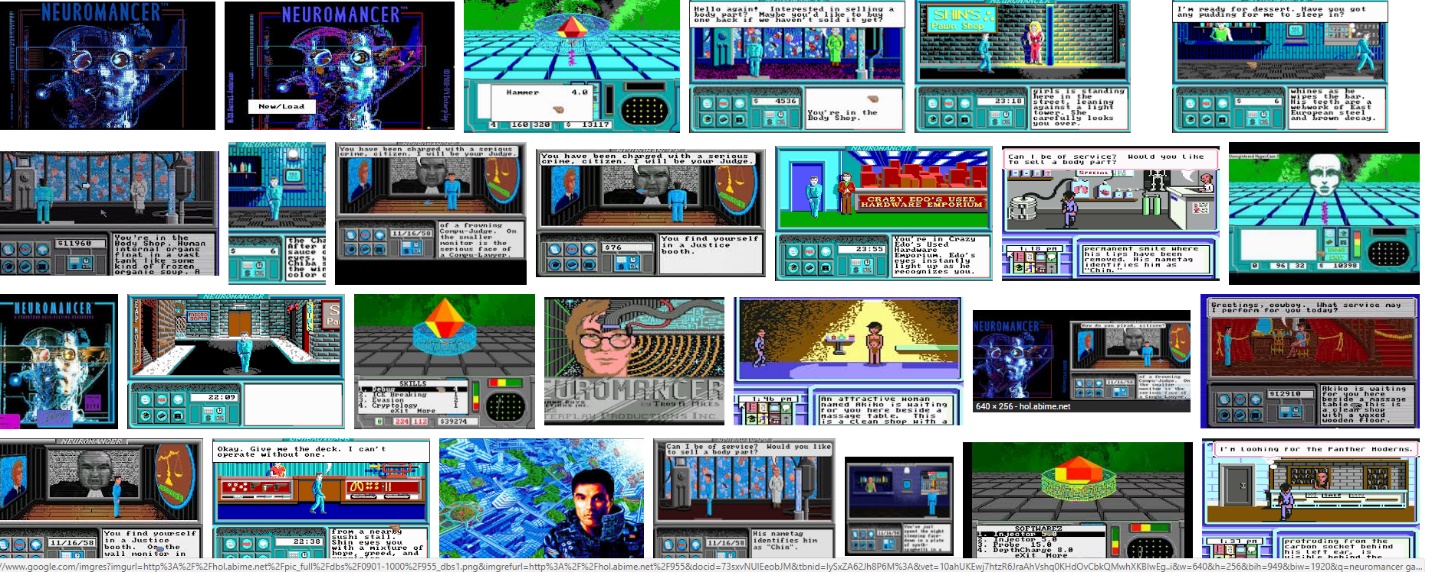
Player will have a menu with the following weapons / abilities:

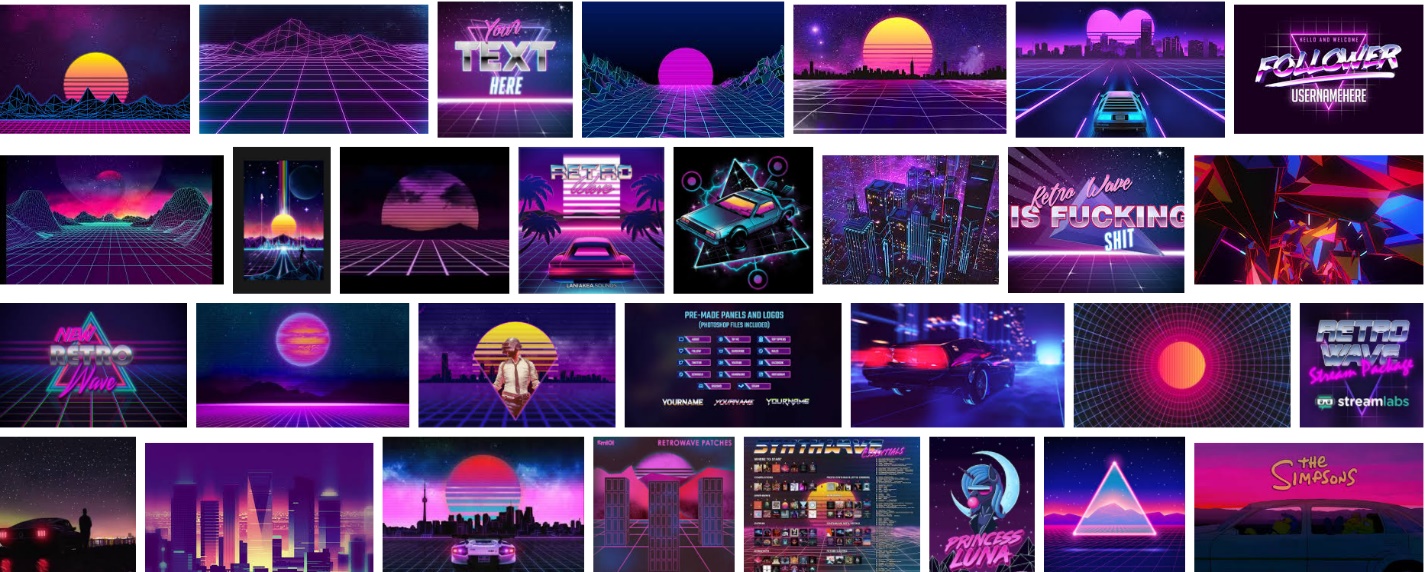
1. ~~DoS Attack (chaingun)~~
2. Data Spike (basic shot)
3. ~~Root Access (like a spell, deactivate enemy and allows you to do some stuff)~~
4. ~~Brute Force (melee)~~
5. ~~Encypt (cloak)~~

## Briefings

~~Bill Gibson will give you mission breifings about getting data caches from things like a 100mb Hitachi Death Star!~~

## Art Style





# Code Standards

1. Main game loop will be handled by a singleton game manager
2. Components will communicate via standard Events and implement custom classes that inherit from EventArgs if necessary
3. Player actions must be abstracted from the invoking controls in order to better support additional devices later in development
4. ~~Procedural generation will only deal with placing level “blocks” and spawnable objects in pre-determined spawn sites in level blocks. This way, we can bake nav meshes, lights, etc.~~
5. No class shall exceed 200 lines of code
6. Each game element will have its own test scene. If necessary, it will also be coupled with unit tests.
7. Game assets will be tied to a build quality setting for future proofing the architecture to support multiple quality settings

# Achievement Roadmap

## Fundamentals (800 total possible, forecasting 600)

* Scale – different ICE should give you a feeling of scale as well as architecture of level “blocks”
* Animation – Items, enemies, and puzzles will probably all include animation
* ~~Lighting – Most lighting should be baked, the alert ICE can probably have a realtime cone. Light probes will be used for ICE as they move~~
* Locomotion – Teleporting with cooldown based on distance traveled
* Physics – Perhaps a puzzle will use physics
* Video – Accomplished by mission brief / tutorials
* Empathy – Will not be attempted

## Completeness (1,500 possible, forecasting 1,000)

* Gamification – Optional items (maybe have different weapons purchasable in store)
* Diegetic UI – Mini game puzzles
* Alternative Storyline – Will not be attempted
* AI – Enemy movement on nav mesh and attack patterns
* 3d Modeling – Enemies and terrain objects modeled in blender
* Photogrammetry – Will not attempt

## Challenges (3,500 possible, forecasting 1,000)

* Speech Recognition – Will not attempt
* User Testing – Mike and Shehzad
* ~~Compute Shader – Amplify Shader for enemies~~
* Mulitplayer – Will not attempt
* Mixed Reality Recording – Will not attempt
* App Store – Maybe steam