Jack Bell
Project E

Woo Hoo! Back to the madness!!

- All the madness! None of the “DIY Engine” jank!

- Featuring:
  - The Python language (*and there was much rejoicing*)
  - Models/Animations that definitely aren’t ripped from WoW
  - Custom Collision!*
  - Custom Events
  - A Puzzle

How do it do?

Python!!!

- YAY! Wait, why is this useful?
  1. I've been using Python for years, so less googling
  2. Python has a slew of pre-built 3D engines (I used Panda3d), so there's now only one or two 'structural' objects I need to program in
  3. There's also a few optimizations that make all this much easier (i.e. passing functions in as parameters, easy optional parameters, no semicolons, etc.)

```
from direct.showbase import DirectObject
from direct.showbase.ShowBase import ShowBase
from direct.task import Task
from direct.actor.Actor import Actor
from direct.interval.IntervalGlobal import Sequence
from panda3d.core import BitMask32
from panda3d.core import DirectionalLight, AmbientLight
from panda3d.core import Point3, CollisionNode, CollisionSphere, Col
```

```
class Interactable():
    #note to prof: Functions are first-class objects in Python, so you
    def __init__(self, name, path, loader, interactFunc, collisionZone,
        self.xPos = xPos
        self.yPos = yPos
        self.zPos = zPos
        self.interactFunc = interactFunc

        self.object = Geom(path, scale, loader, xPos, yPos, zPos)

        triggerZone.tangible = False
        tColliderNode = CollisionNode(name)
        tColliderNode.addSolid(triggerZone)
        self.tCollider = self.object.model.attachNewNode(tColliderNode)
```

```
class Geom():
    #String Path: Path to model, Array scale: x,y,z amount to sca
    def __init__(self, path, scale, loader, xPos=0, yPos=0, zPos=
        self.xPos = xPos
        self.yPos = yPos
        self.zPos = zPos

        self.model = loader.loadModel(path)
```

1000% Original models and animations

- Ok ok, *some* I did actually make
  1. Panda3d has a custom 3D file format (supports baked textures and animations)
  2. Thankfully, there's a pre-made CLI (Command Line Interface) for converting directly from blender files!
  3. Once the file/model is converted, load it into the program and reparent it to a "PandaNode" to apply game logic to it

```
# #Cave
cave = Geom("models/cave.bam", [.6], self.loader)
caveColliderNode = CollisionNode("caveSphere")
caveSphere = caveColliderNode.addSolid(CollisionInvSphere(0,0,0,18))
caveSphereCollider = cave.model.attachNewNode(caveColliderNode)
cave.render(self.render)
# ##Player##
#add a animated/moving model to the scene
playerAnims = {"walk": "models/player/walk.bam", "idle": "models/player/idle.bam", "danc
self.playerActor = Actor("models/player/player.bam", playerAnims)
self.playerActor.setScale(1,1,1)
self.playerActor.setPos(0,0,0)
self.playerActor.reparentTo(self.render)

playerColliderNode = CollisionNode("player")
playerColliderNode.addSolid(CollisionCapsule(0,0,.6,0,0,.4,.3))
playerCollider = self.playerActor.attachNewNode(playerColliderNode)
#playerCollider.show()

self.pusher.addCollider(playerCollider, self.playerActor)
self.cTrav.addCollider(playerCollider, self.pusher)
```

Custom Collision Meshes!*

- *Technically possible, but really buggy
  1. This is defined in the model file using the '<Collide>' flag
  2. The issue is that doing this makes the collision mesh only only one polygon thick (i.e. smaller than the player hitbox, thus allowing the player to easily bypass the collision)
  3. Better, but less flexible solution is to use predefined 'Collision Solids'

```
    <RGBA> { 1 1 1 1 }
  }
}
<Group> Cave.001 {
<Collide> CaveCol { polyset descend keep }
  <Transform> {
    <Scale> { 10.3711223602295 10.3711223602295 -10.371122
    <Rotate> { 180.000005008956 0 0 -1 }
    <Translate> { 0 0 1.96696090698242 }
  }
  <Group> ReverseCulling {
```

```
caveColliderNode = CollisionNode("caveSphere")
caveSphere = caveColliderNode.addSolid(CollisionInvSphere(0,0,0,18))
caveSphereCollider = cave.model.attachNewNode(caveColliderNode)
cave.render(self.render)

# #Statue
statueColZone = CollisionCapsule(0,0,100,0,0,-100,30)
statueTZone = CollisionCapsule(0,0,100,0,0,-100,70)
self.statue = Interactable("statue", "models/interactables/statue.bam", self.load

playerColliderNode = CollisionNode("player")
playerColliderNode.addSolid(CollisionCapsule(0,0,.6,0,0,.4,.3))
playerCollider = self.playerActor.attachNewNode(playerColliderNode)
#playerCollider.show()
```

Custom Events

- Very simple, but sort of involved
  1. Define a custom event handler class
  2. Declare bool vars for each event
  3. Define what action should happen when an event is called
  4. Define said actions themselves
  5. Listen for those events inside the main loop

```
#mostly just gets key strokes and updates v
class Events(DirectObject.DirectObject):
    def __init__(self):
        # ## Internal Vars
        self.e = False
        self.q = False
        self.p = False

        self.w = False
        self.a = False
        self.s = False
        self.d = False
        self.space = False

        self.accept('e', self.ePressed)
        self.accept('e-up', self.eDepressed)

        self.accept('q', self.qPressed)
        self.accept('q-up', self.qDepressed)

        self.accept('r', self.rPressed)
        self.accept('r-up', self.rDepressed)

        self.accept('w', self.wPressed)
        self.accept('w-up', self.wDepressed)

        self.accept('a', self.aPressed)
        self.accept('a-up', self.aDepressed)

        self.accept('s', self.sPressed)
        self.accept('s-up', self.sDepressed)
```

```
    def ePressed(self):
        self.e = True
    def eDepressed(self):
        self.e = False

    def qPressed(self):
        self.q = True
    def qDepressed(self):
        self.q = False

    def wPressed(self):
        self.w = True
    def wDepressed(self):
        self.w = False
```

```
if events.w:
    self.move('w')
if events.a:
    self.move('a')
if events.s:
    self.move('s')
if events.d:
    self.move('d')
if events.space:
    self.move(' ')
```

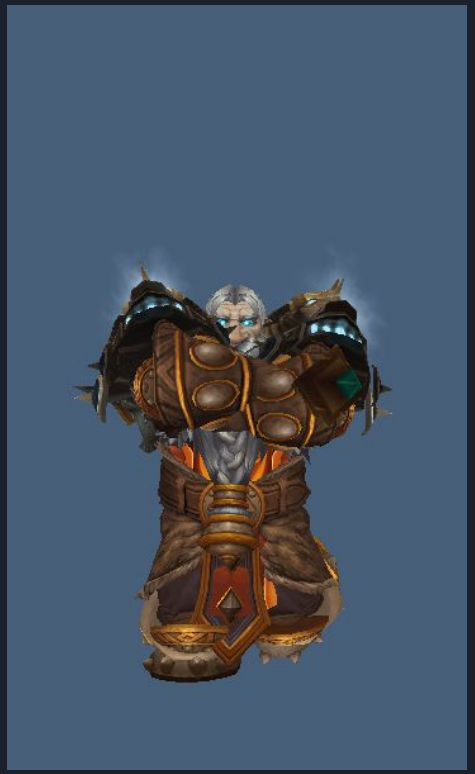
Puzzle Logic

- Much Dynamic, Many wow
  1. Using the event logic, define an “interactable” object that takes an “interaction function” as a parameter
  2. Listen for the ‘x’ key in the mainloop, and trigger the interactable’s interaction function when the player presses ‘x’ and is in the interactable’s “trigger zone”
  3. If the player interacts with the statue, rotate it to the next position -- based on what side of the statue the player is on -- using a preset array of rotational positions
  4. Based on the position of the statue, add a color -- what waterfall the statue is facing -- to an array
  5. If that array == length 3, see if the array matches the solution array, if yes, you win!

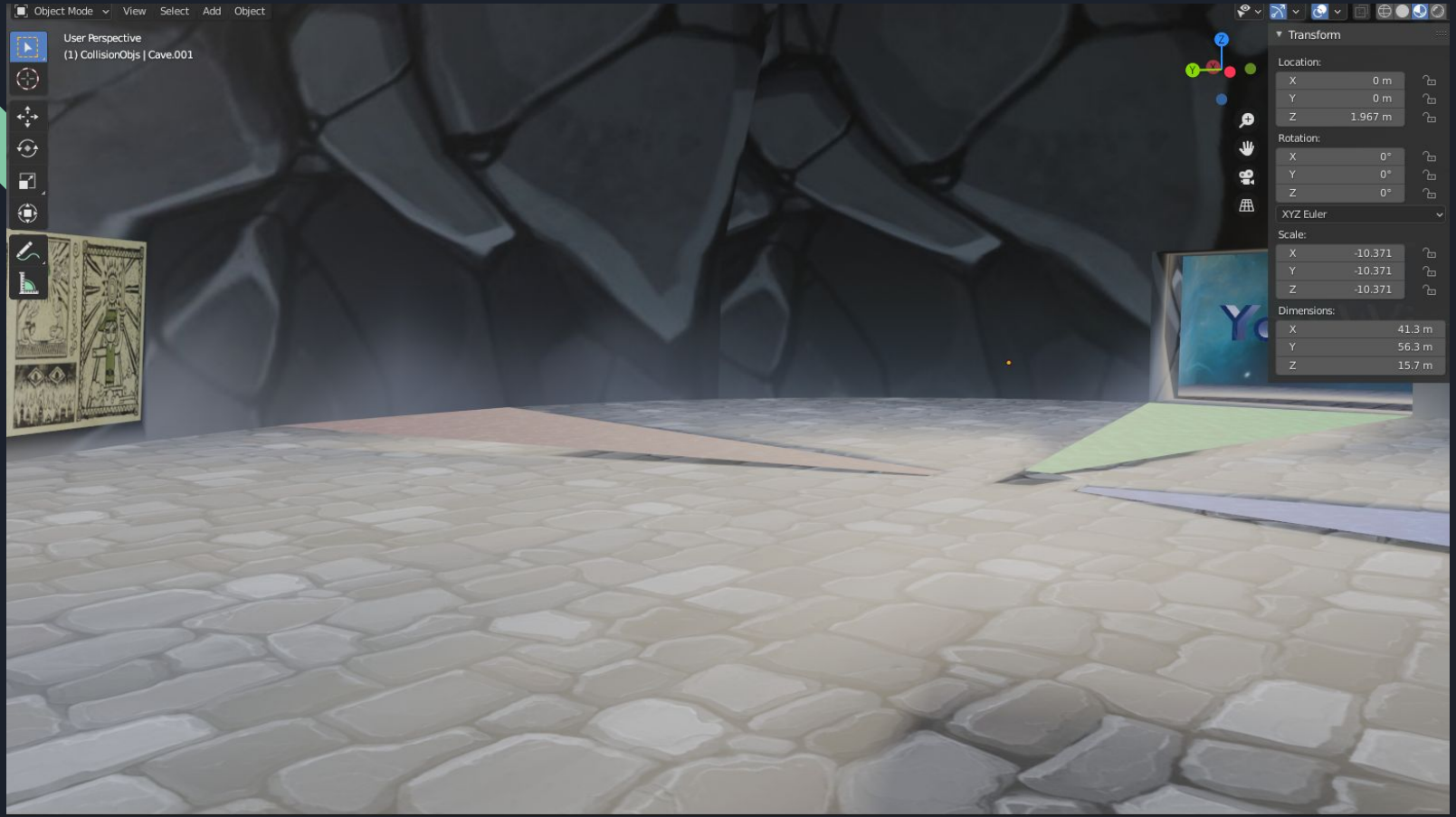
```
def Statue(self, statue, directon):
    print("started at", self.statueIncrimenter)
    if (statue.object.model.getH() == -190) and (self.statueIncrimenter == 0) and (directon == "forwards"):
        self.statueIncrimenter = 0
    elif self.statueIncrimenter == 0:
        if directon == "forwards":
            self.statueIncrimenter +=1
        elif directon == "backwards":
            self.statueIncrimenter = 2
    elif self.statueIncrimenter == 1:
        if directon == "forwards":
            self.statueIncrimenter +=1
        elif directon == "backwards":
            self.statueIncrimenter -= 1
    elif self.statueIncrimenter == 2:
        if directon == "forwards":
            self.statueIncrimenter = 0
        elif directon == "backwards":
            self.statueIncrimenter -= 1

    print("ended at", self.statueIncrimenter)

    if self.statueIncrimenter == 0:
        statue.object.model.setH(self.statuePositions[0])
        self.statueLookedAt.append("blue")
    elif self.statueIncrimenter == 1:
        statue.object.model.setH(self.statuePositions[1])
        self.statueLookedAt.append("green")
    elif self.statueIncrimenter == 2:
        statue.object.model.setH(self.statuePositions[2])
        self.statueLookedAt.append("red")

    print("Statue Rotated")
    print(self.statueLookedAt, "next should be", self.statueIncrimenter)

    if len(self.statueLookedAt) >= 3:
        if self.statueLookedAt == ["blue", "red", "green"]:
            return True
        else:
            self.statueIncrimenter = 0
            statue.object.model.setH(-190)
            self.statueLookedAt = []
```

```
if events.x:
    if events.inStatueZone:
        print("Player X =", self.playerActor.getX())
        if self.playerActor.getX() < 0:
            if self.statue.interactFunc(self.statue, "forwards"):
                messenger.send("solved")
        elif self.playerActor.getX() > 0:
            if self.statue.interactFunc(self.statue, "backwards"):
                messenger.send("solved")

        events.x = False

if events.solved:
    self.waterfallGreen.object.model.setPos(6,-8,200)
```

Mood Board/”Sketches”



Where I got the models for animation/exporting them



Creating the static models

Part 1: A Near Death Experience

Players start out at the entrance to The Cave of Forgotten, having just set up camp outside and arriving there from Khar'ezor.

In the Dwarven capital of Khar'ezor, the somewhat deranged wizard Fulminis Braccae has hired each of you to help him retrieve a legendary sword known as 'Evil's Bane.'
Have players say how they met Fulminis
Over the past 3 days, he has led you to an ancient place which he refers to as "The cave of Forgotten." In your travels with him, he reveals that he believes this sword to be part of an ancient prophecy about uniting the magic of old with the magic of new and wants the artifact to research.

Your party has just arrived at a somewhat large hill, where Fulminis instructed you to set up camp. While you were doing that, Fulminis was running frantically about the hill, carrying some sort of map, muttering inaudibly to himself as he looked between the hill and the map.

The Party's camp contains enough shovels and pickaxes for each of them, the cart and two horses they brought, enough provisions for each of them to last a week, 3 two person tents and one one-person tent for Fulminis, and coin totalling 50gp.

The story begins with the party just finishing pitching camp, and Fulminis frantically shouting that he's found "it" and asking the party to join him on the other side of the hill.

When the players arrive at the other side of the hill, Fulminis gestures excitedly to a random part of the hill and tells them that this is the entrance they've been looking for! Based on the parties' reaction, he tells them that, based on the age of his map, the entrance is here, just ~50-100 feet below them, so get digging! He then asks if they brought the digging supplies, if they did he asks for his shovel and pick, if not he tells them to go get them, unless they want to join him in the old fashioned way and starts digging in the dirt with his hands.

After the players dig ~25ft down, they uncover a set of stone doors leading into a natural, mineshaft-esc, hallway, which itself leads into The Main Chamber.

The Main Chamber:

When players begin to enter the main chamber of the cave, have them make a perception check and read the corresponding description:

No Darkvision or <5:

the ground appears to be more worn and cracked than the rest of the chamber's floor, almost as if the statue has been sloppily rotated, scraping up the ground. Around that is a small mass of land, and around that; a shallow moat of clear water, which appears to be filled by three, colored water falls on the Eastern (red), Southern (green), and Western (blue) walls of the room.

nat. 20:
As you peer into the dark, crudely circular chamber, you feel a cool air hit you, accompanied by the not unpleasant smell of flowing water. As you look around the room, which you determine to be 102 feet in diameter and 83 ft high, you notice a mural drawn around the back of the room (the part you entered), which depicts some sort of story. You do find it odd that there appear to be vibrant colors on the mural, despite it looking centuries old. Looking deeper into the chamber, you see a statue of a knight holding his sword pointed at the ground, all of which seems oddly pristine. Looking between the statue and the mural, you notice that the sword held by the statue and the sword held by one of the figures in the mural are the same. Directly around the statue, the ground appears to be more worn and cracked than the rest of the chamber's floor, almost as if the statue has been sloppily rotated, scraping up the ground. Around that is a small mass of land, and around that; a shallow moat of clear water, which appears to be filled by three, colored water falls on the Eastern (red), Southern (green), and Western (blue) walls of the room, the colors of which directly match the colors of the flames on the mural.

Walking up to the waterfalls also reveals that they are colored. If no player notices the mural, Fulminis will call it to the party's attention. (Basically, use Fulminis to drop progressively more obvious hints if the party seems stuck).

Investigating the statue also reveals the worn ground, and high rolls (DC 15+) reveal that the sword is the same as the one weilded by the hero in the mural. Same for the Red and Blue waterfalls (DC 15 to determine similarities) However, for the Green waterfall, a DC 15 investigation reveals the color similarity, but a DC 20 Investigation check reveals a hidden door behind the waterfall (The players cannot then open this like a normal door, though it could be destroyed by a great force, aside from solving the puzzle).

Casting Detect Magic on/near respective objects reveals that the Statue, Mural, and waterfalls have all been enchanted with a permanent form of Prestidigitation to create their odd effects. If a player casts the spell while standing next to the green waterfall, it also reveals that there are two, powerful magical objects sitting just 30ft behind the waterfall (*Frigryr* and *Karhir Vaktinar)*.

If the party asks, Fulminis can get the gist of the mural (written in Hytothian), but does not know the full translation. The gist is as follows: The people of Hythos were given divine inspiration and prospered with revolutionary inventions. One day, a great evil stole godly power and brought the kingdom to its knees. Until a brave hero found, within himself, hallowed valor and vanquished the evil one. With the kingdom saved; his purpose fulfilled, the hero left Hythos seeking new adventure. It is fated that the evil shall return in his absence and lay the kingdom to waste once more. When the three celestial powers are brought together, the hero of old shall grant his weapon to the hero of new.

The Puzzle:

The party is expected to determine that the statue in the chamber a) rotates and b) is a representation of the hero. They are then expected to combine this with the fact that the hero is green and that he carries the green flame (**Green Flame!**) and that the waterfalls are representative of the three flames.

The solution is to rotate the statue to face the waterfalls in the order their colors appear (I.E. Blue, Red, Green) doing so opens the secret door and splits the waterfall, creating a path through the water on the ground (I.E. Waterfall splits, diverts water, dries center strip, leading to Altar Room).

The Altar Room:

As the party enters this room, read the following:

Walking into the room, which is still clearly very old, but is in a much better condition than the previous chamber, you notice three suits of armor standing facing inwards on both sides of the room. At the end of the room is an altar of sorts, made of a crumbling marble and covered in a once royal red cloth, now tattered and torn. Atop which, with their opening facing you, sits a set of intricately crafted gauntlets made of a faded, bronze-like metal, with some sort of blue crystal inlaid into the back of the right hand. In the grasp of these ancient gauntlets, lies a warhammer, about 3 ft long, of equally legendary craftsmanship, however seemingly much older. Despite this, the inside of the hammer's head seems to be glowing a powerful orange as if containing the might of a star within it.
When he enters from behind you all, Fulminis says "Well, this is not what I

The OG source text

The Fall:

Long ago, there existed a highly advanced kingdom. Long ago, their peace was kept by the Brave Loin and Wise Zeulas.

But long ago, the Eldritch of great evil and madness tricked Brave Loin and killed him. Without its protector, the Treacherous One spread ruin and madness throughout the land, and when the final hour seemed at hand...

...Brave Zeulas returned from the Land Beyond. Wielding the Staff of the Great Wizard Yock, she released the power from the staff, banishing that Eldritch for a time.

With haste, that kingdom set to seal their armaments away and prepare the tests for Loin's return. Soon after, that Eldritch returned, having freed his fellows.

The people hoped that the Protector of Hythos would return from the Land Beyond... But there he remained. Laughing at their folly, the Eldritch began their onslaught, removing the kingdom from the world.

To lift -- Rauuv
To change, modify -- Altuuv
To grapple, hold on to -- Huudoonuv
To protect, defend -- Problaasuv
To face (i.e. in a duel) -- Abruuz

Nouns:

-Nouns come in 3 declensions (1st being for everyday words, 2nd being for Magical words, and 3rd being for techy words) and have 3 genders (M(he), F(she), and N(it))
-for negations: ne ____ pas
-each declension is broken up into the standard Latinate things
-More often than not, for nominative singular, nouns will have no ending

Noun Endings:
Nominative: Denotes the subject or main thing of the sentence. (Normally, a word will use it's 'normal' ending when it's the subject, and the '-uum' for when it's referring to something specific but with a normal noun; i.e. 'The Lord,' or 'The King,' or (rarely) 'That certain ___')
Genitive: Denotes possession (of ___/___'s)
Dative: Denotes the direct object (the thing the verb is happening to)
Accusative: Denotes the indirect object (thing that the Nominative is using to do the verb, kinda sorta)
Ablative: Denotes relative location (in/on/under/above etc.)
Vocative: Denotes a thing being commanded

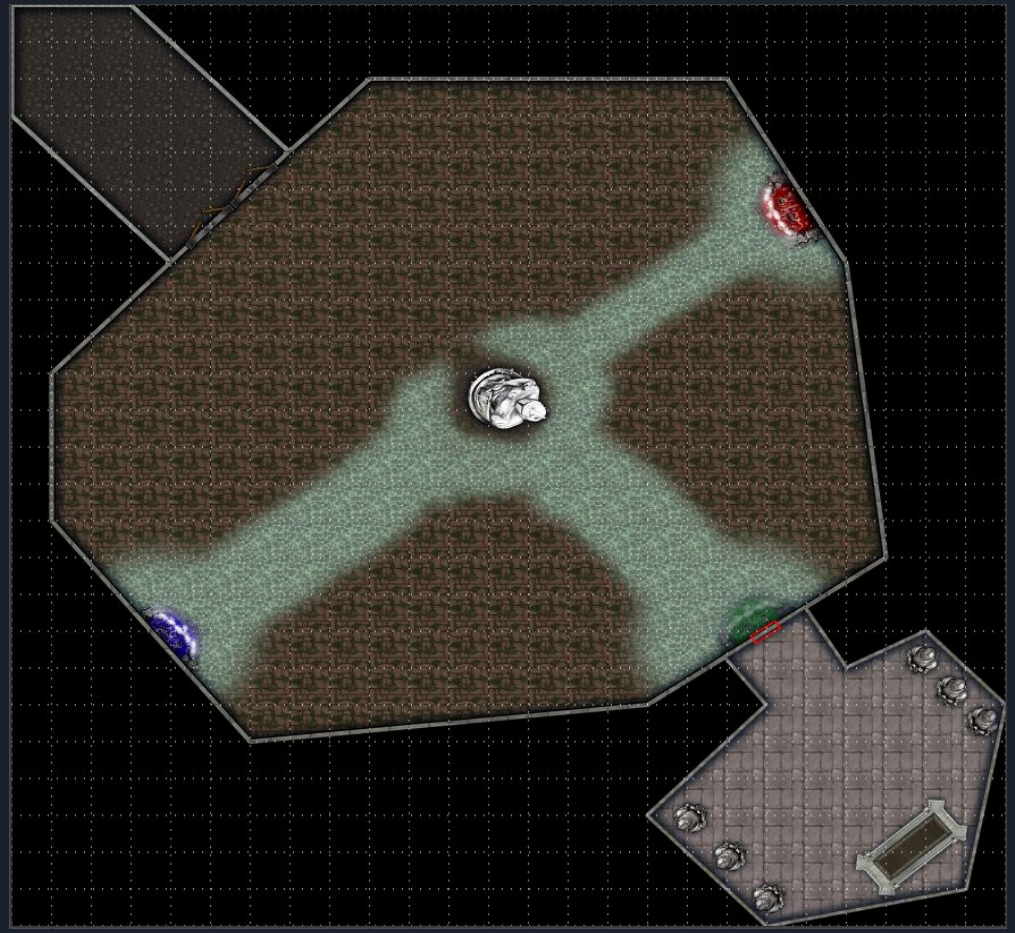
1st (every day)	Sing	Plural
Nom	-uum/ -__	-uums
Gen	-as	-a
Dat	-im	-ism
Acc	-am	-sam
Abl	-uus	-ud
Voc	-uu	-ii

2nd (Magi	Sing	Plural

Patemp tuullim, uusentreim relythas chanlilas huuotas. Patemp tuullim, nuusnas pav uusememuuvim Loinuus Liinuus ent Zeuluus Rendiuus.

Mee patemp tuullim, Malyakov Malnolo Moguunolent Insuunas edgoruuzdid Loinuus Liinuus ent edfuunavdid thram. Navas praehatn nimas, Malyakov Insuunas insruuruuzdid Withrament Insuunem relythuus.

Writing/Translating the story on the wall



The OG dungeon map

Final

Final Thoughts

To be honest, I didn't learn a lot from the class. This was expected Given that I teach coding professionally! That being said, while I didn't learn anything per-se, I loved being able to challenge myself by going well above the assignment, allowing me to sharpen my skills (particularly in thinking about events handling and navigating 3d space). I immensely enjoyed the class, the opportunity to challenge myself, the opportunity to teach all of you, and getting to know all of you! Y'all are a great bunch, and I truly hope our paths cross again!