(Auto) Avaliação do PI

Relatório de acompanhamento do semestre letivo afetado pela pandemia.

Nome: *Wellington Maciel cordeiro de Elizeu*

Nome do grupo: *Wellington Maciel cordeiro de Elizeu*

Nome do jogo: *Apollo 20 - The Return*

Desafios do Projeto Integrador:

1) Implementar feedback dos propulsores:
   a) Um único sprite que sinalize feedback quando os propulsores estiverem ligados (Sprite diferente do código base, e com posição dos propulsores em lugares diferentes)**: 0,5 pts**
   b) Animação de sprites (Sprite diferente do código base, e com posição dos propulsores em lugares diferentes): **1,0 pts**
   c) Polígono em formato de triângulo que varie proporcionalmente com o valor da propulsão usando diretamente o valor da variável da propulsão (Sprite diferente do código base, e com posição dos propulsores em lugares diferentes). **2 pts**.

2) Avaliação do pouso da nave
   a) Pousar em qualquer lugar do "chão" da lua. **0,5 pts**
   b) Detectar três tipos pouso: 1) perfeito - se foi perpendicular ao "chão" da lua com uma variação angular mínima; 2) variação angular máxima para pouso aceitável resultando em pouso imperfeito; 3) caso contrário a nave explode. **1,0 pts.**

   c) Criar locais específicos de pouso, variando no eixo X e Y, além de implementar tudo do item b). **2 pts**

3) Feedback sonoro
   a) Efeitos sonoros básicos do propulsor do foguete, pouso e explosão **0,5 pts**
   b) Música de fundo **0,5 pts**

4) Score/Pontuação e ranking
   a) Criar sistema de pontuação que calcule a somatória de pontos de mais de 1 característica do game como por exemplo: quanto menor tempo maior pontuação, quanto menos gasolina gasta, maior pontuação, pontuação maior para pouso perfeito, menor para pouso aceitável. **0,5 pts**
   b) Criar sistema de ranking que aparece ao final do jogo, para isso será necessário solicitar ao jogador um nome ou abreviatura do nome. **0,5 pts**
   c) Salvar o ranking num arquivo texto externo e carregar ele no início do jogo **0,5 pts**

5) Fases e desafios
   a) Criar pelo menos 3 fases **0,5 pts**
   b) Criar fase procedural/aleatória, onde o terreno e local de pouso são diferentes a cada vez que o jogo é inicializado **2,0 pts**
   c) Criar coletáveis, por exemplo: itens que aumentam a pontuação, gasolina, ou algum tipo de item que seja obrigatório para a missão **0,5 pts** cada (consultar com o professor para ter certeza que o coletável irá somar pontos)

6) HUD
    a) Criar HUD simples apenas com texto e números dos dados da nave e jogo **0,5 pts**
    b) Criar HUD mais complexa com texto, barras de status e imagem/ícone das informações da nave e jogo **1,0 pts**

7) Menu
    a) Menu simples apenas com “Start Game” e “Quit” **0,5 pts**
    b) Opções onde seja possível ajustar pelo menos áudio **0,5 pts**
    c) Possibilidade de mudar as teclas do teclado para controlar a nave **0,5 pts**
    d) Adicionar no menu “Créditos” e “Ranking” **0,5 pts**

8) Apresentação final mais trailer de 2 minutos no máximo **1,0 pts**

Sob seu ponto de vista, explique como o ensino remoto e a pandemia influenciou o desenvolvimento das atividades individuais e do grupo.

Consegui ver pontos positivos em meu desta quarentena, o principal foi poder rever as aulas fornecidas pelos professores para posteriormente consulta. Nem toda duvida surge no ato da aula.

Me senti prejudicado em relação ao hardware pois ficamos sem acesso as maquinas do Senac e nem todos possui um bom computador.

Cite cada um dos desafios citados acima sua equipe escolheu para desenvolver em seu jogo:

1) Implementar feedback de dois propulsores

c) Dois propulsores que variam de tamanho conforme aceleração da nave. ***2 pts****.*

2) implementar 3 tipos de pouso:

b) Perfeito - se foi perpendicular a área de pouso situado na lua com uma variação angular mínima;

Tolerável - se foi com uma variação média tolerável

Forçado - variação angular máxima para pouso aceitável resultando em pouso imperfeito; 3) caso contrário a nave explode. ***1,0 pts.***

2) Criar locais específicos de pouso:

c) variando no eixo X e Y, além de implementar tudo do item b). ***2 pts****.*

3) Feedback sonoro:

a) Efeitos sonoros básicos do propulsor do foguete, pouso e explosão ***0,5 pts***

5) Fases e desafios:

b) Criar fase procedural/aleatória, onde o terreno e local de pouso são diferentes a cada vez que o jogo é inicializado ***2 pts****.*

c) Criar coletáveis, por exemplo: itens que aumentam a pontuação, gasolina, ou algum tipo de item que seja obrigatório para a missão ***0,5 pts***

6) HUD:

b) Criar HUD mais complexa com texto, barras de status e imagem/ícone das informações da nave e jogo **1,0 pts.**

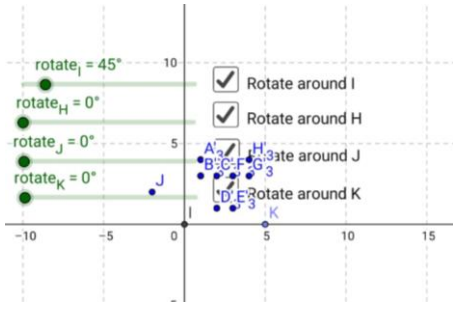
7) Menu:

a) Menu simples apenas com “Start Game” e “Quit” **0,5 pts**

b) Opções onde seja possível ajustar pelo menos áudio **0,5 pts**

Dificuldade e resoluções de problemas durante o processo de desenvolvimento

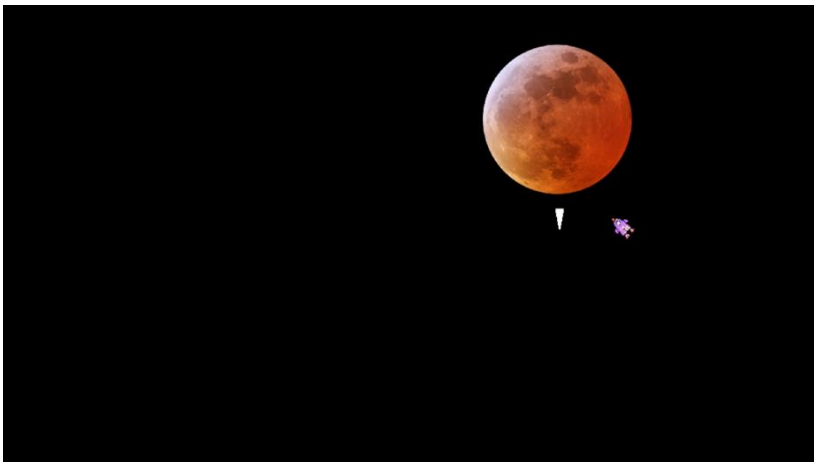
Primeiro desenvolvi a movimentação da nave e o próximo objetivo era criar dois propulsores, tive a percepção de que eu deveria tentar fazer apenas um e se desse certo teria sucesso em replicar para que se tornasse dois propulsores.



Umas das maiores dificuldades que eu tive foi fazer com que os polígonos

rotacionassem de acordo com a rotação da nave, foi preciso fazer vários estudo paralelos sobre Geometria analítica, e também o site Geogebra foi de grande utilidade para verificar algumas funções vetoriais e também poder testa-las. Mesmo assim ainda foi difícil pelo fato de ainda estar se familiarizando com a biblioteca Pygame, muita das vezes a minha lógica já estava correta, mas não conseguia implementa-la de primeira

instancia. Outro motivo da dificuldade foi porque essa etapa foi uma das primeiras tarefas a ser feitas no projeto logo após o desenvolvimento da movimentação da nave que usa Sprite, e não tivemos conhecimentos prévios necessários de como lidar com polígonos.

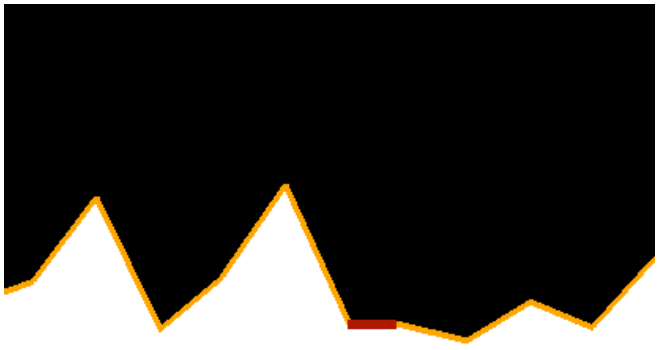


Um segundo desafio que deu bastante trabalho foi a construção do terreno ser procedural, porem a construção do gráfico (forma do polígono) deu menos trabalho pois já havia ganhado um pouco mais de experiencia na construção dos propulsores.

O resultado dessa imagem abaixo já estava quase chegando em um resultado esperado satisfatório, antes o terreno já estava sendo gerado proceduralmente, porem ainda não me satisfazia porque estava sorteando apenas duas alturas mínima e máxima possíveis para que o fosse sorteado entre elas, dando uma aparência de "zigue-zague" então revisei o código e consegui fazer um "Randon dos randons" ou seja, a classe sortear duas possíveis alturas de terreno, uma cobrindo 20% da tela e outra com 30%, depois que foi sorteado uma das porcentagem, é sorteado novamente dentre essa porcentagem um valor mínimo e máximo dessa porcentagem, isso tudo em no mesmo ciclo de desenho e assim foi possível construir terrenos que possuía não apenas duas linhas do horizonte sendo mínimo e máximo, foi possível criar 4 linhas horizontais de mínimo e máximo de altura.

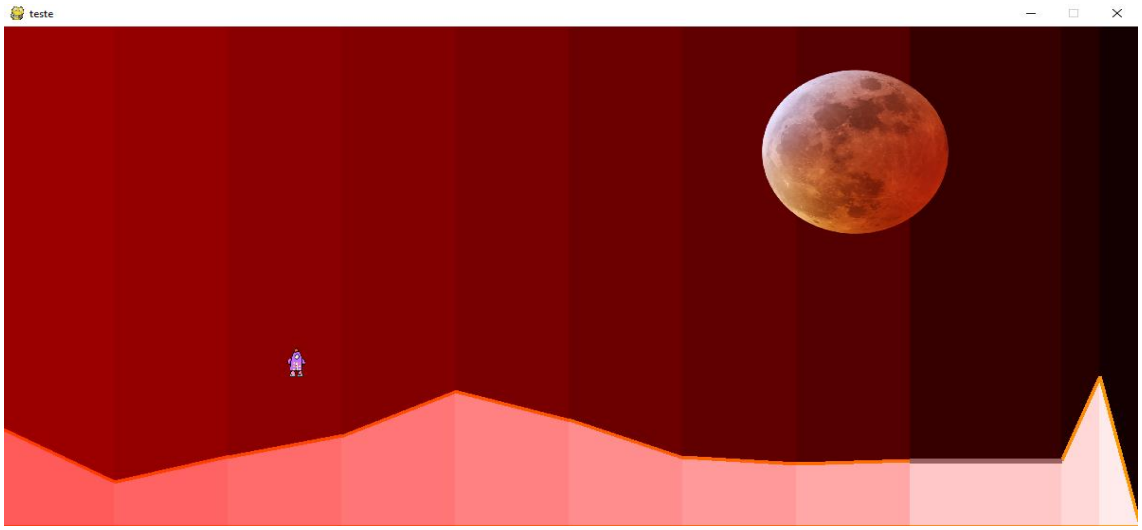
A partir desse momento foi possível construir um terreno que tivesse variações mais distintas.

A criação procedural da área de pouso é sorteada através do contador que gerencia quantas divisões vai ter na horizontal da tela, ao aumentar a quantidade de tuplas do terreno o código se adapta a essa nova informação e é desenhado um terreno com mais ou menos informação. Quando um contador é sorteado ele calcula qual é o centro de um polígono ao entre o atual e o próximo é aí que será inserido a área de pouso que

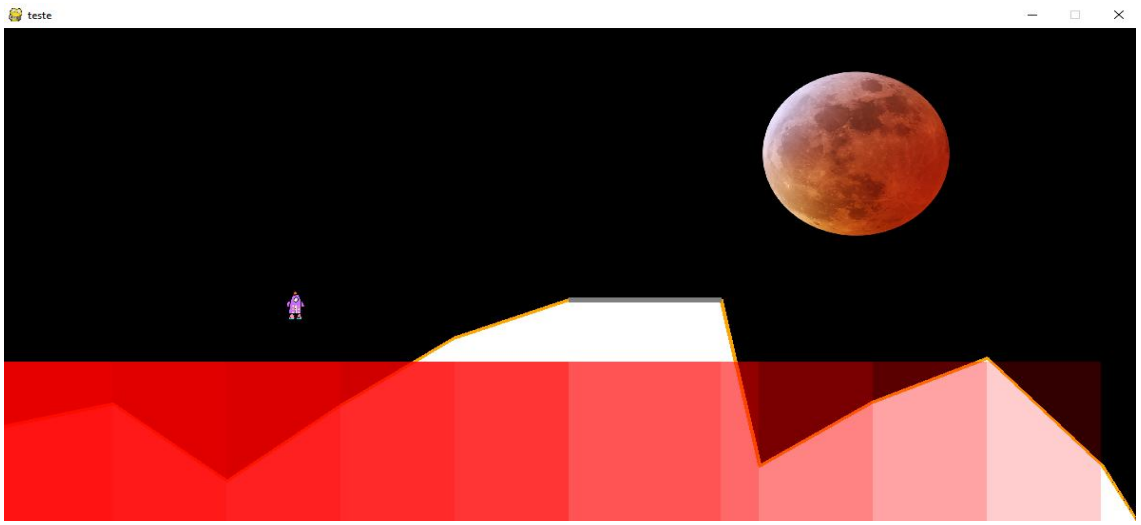


também é levada em consideração o tamanho da nave atual, ressaltando que o terreno também é gerado dinamicamente em qualquer resolução disponibilizada.

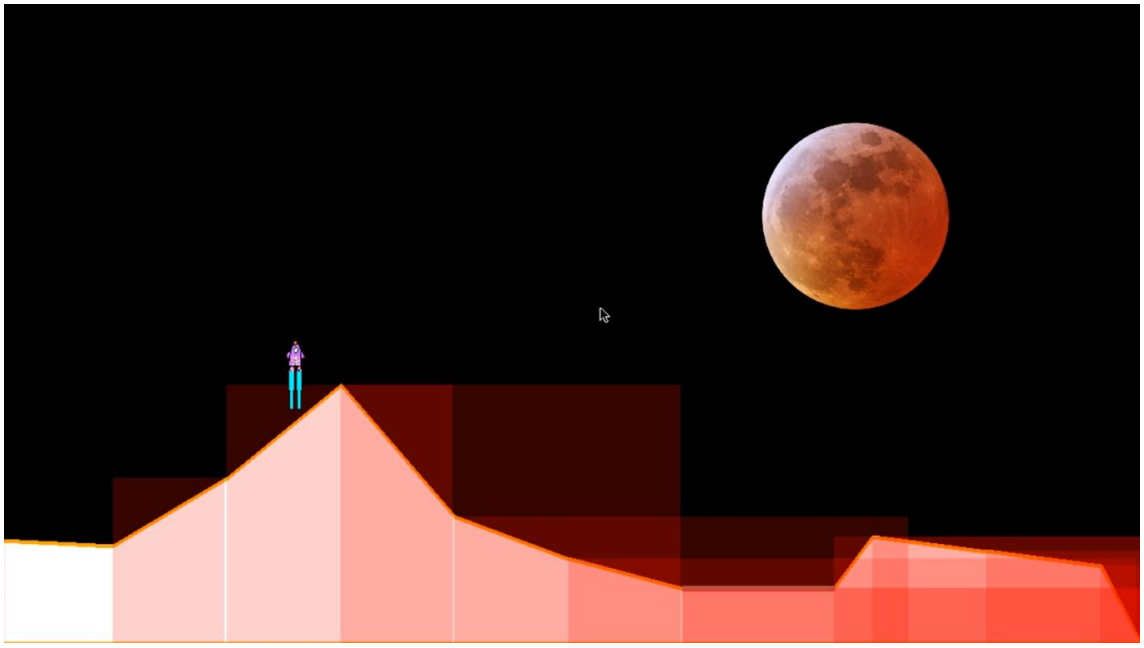
Agora logo abaixo, já pesava que os maiores desafios já tivesse sido resolvido, até chegar na programação das colisoes com o terreno, pois em tudo que eu faço eu quero ou pelomenos tento obter os melhores resultados, logo abaixo, nesta imagem, foi um dos primeiros testes de uma logica que me veio na cabeça para efetuar a construcao das colisoes. Eu pensei... bom se o terreno já tem uma abstracao de uma matrix para ser costruido eu posso gerar os retangulos e capturar não a colisao do terreno em si, mas sim a area de colisao desses retangulos que eu criei. Até essa fase de teste foi um sucesso. Foi preciso fazer retangulos com controle de opacidade para que eu pudesse receber um melhor feedback.

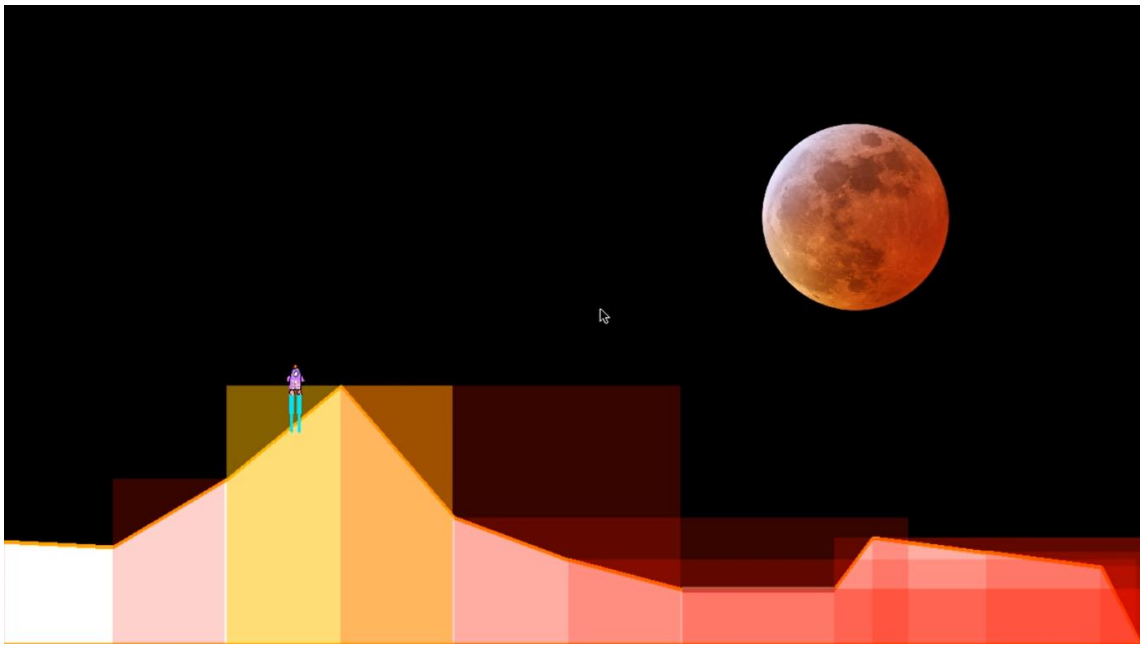


O próximo passo era então controlar esses retângulos para que cada um recebesse a altura do terreno gerado proceduralmente.



A intecão era criar a primeira instancia retangulos que fossem gerados a a partir de um vertice “pai” da esquerda pra direita ate o proximo vertice, e tambem fazer com que a altura do retangulo indepedente do vertice pai ou do proximo ser o mais alto o ele sempre receberia a altura maior entre os dois, isso iria definir a colisao desse determinado trecho do terreno, essa etapa deu um certo trabalho mais tambem foi um sucesso. Implementei um sistema que quando a nave colidia com esse triangulo eu recebia o feedback desse triangulo mudando a opacidade do mesmo.



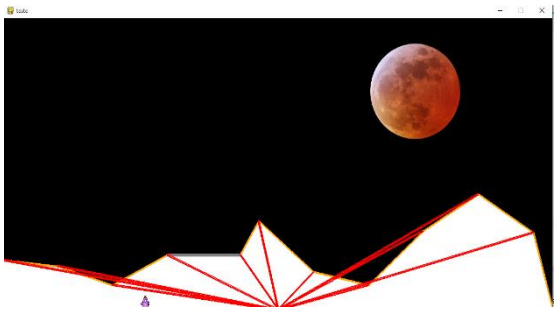


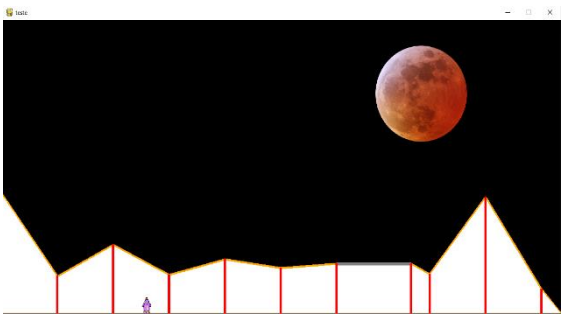
Aí que surgiram as dores de cabeça, eu precisava redividir esses retângulos e compor novos retângulos recebendo divisões na horizontal e na vertical de acordo com seu espaço delimitado conforme a imagem acima, isso iria fazer as quinas dos retângulos menores se aproximar melhor como se fosse uma “escadinha de retângulos colisores” e aproximar mais essa colisão retangular

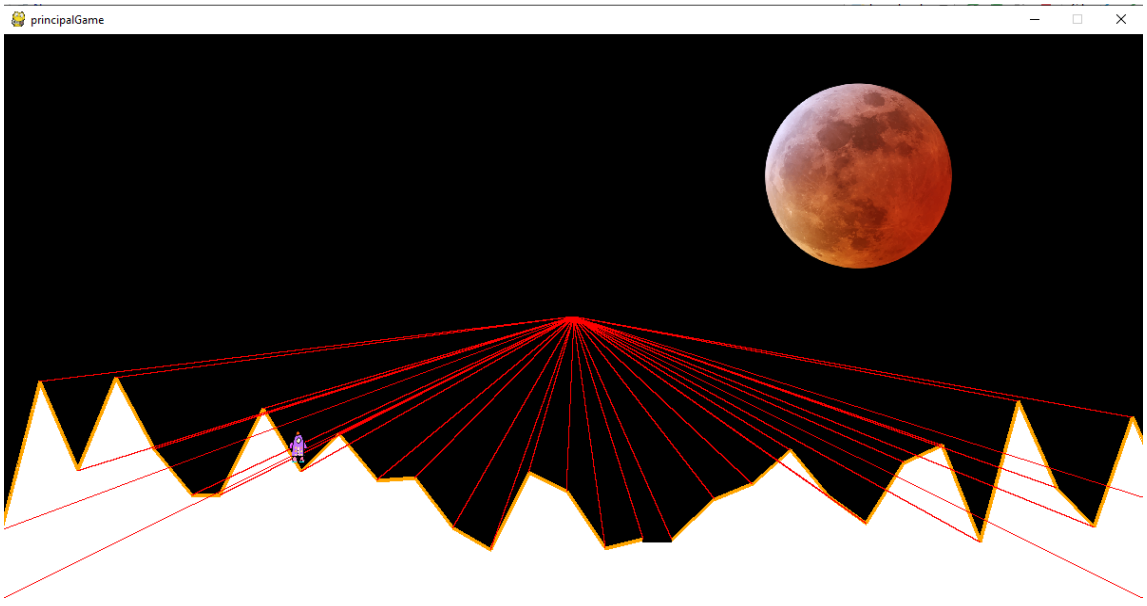
com a forma do terreno. Infelizmente nessa parte do projeto me esqueci de printar (: Mas o que posso dizer é que foi um fracasso, pois queria obter uma colisão mais perfeita e o que eu tinha em mente de fazer a escadinha não consegui implementar de forma aceitável, do meu ponto de vista.

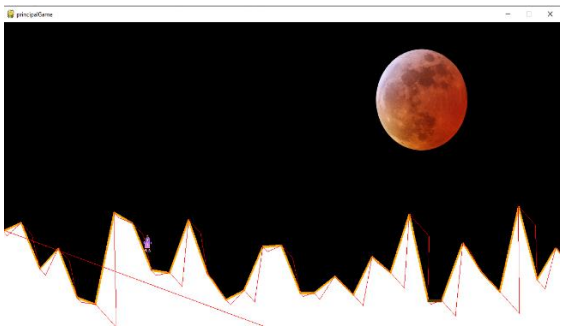
De certa forma frustrado com o trabalho perdido um certo dia me veio na cabeça um programa que eu fiz no 2° ou 3° semestre da faculdade, falando de uma forma geral um sistema central de corpo de bombeiros em que ele deveria identificar a base do distrito mais próximo para atender o chamado de socorro. Eu fiz isso usando um sistema que calculava a distância entre um ponto_xy até outro ponto_xy. Isso foi a luz que me fez sonhar em um sistema de colisão perfeito para a minha necessidade. A partir daí eu pensei em uma lógica.... bom se o terreno é feito com pygame que se assemelha ao plano cartesiano e o terreno possui pontos (vértices com localização nesse espaço) então porque não criar mais um ponto na localização inferior central da nave em que eu possa dentro de um loop verificar a distância do ponto da nave até os pontos do terreno?!

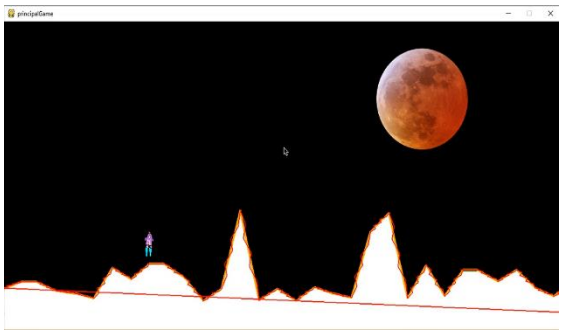
Pra isso ser possível eu precisei criar novos pontos entre os pontos existentes pois senão ficaria buracos inalcançáveis pela condição de distância que dispara a colisão, se eu não fizesse isso eu teria dois problemas, um seria deixar a condição de distância muito alta, isso impediria da nave passar entre dois pontos sem que fosse detectada, mas por outro lado quando a nave caísse de cima para baixo na direção de um ponto do terreno ela detectaria colisão com uma distância muito alta. Por isso crie novos pontos. Criar esses novos pontos que acompanhasse o contorno do terreno randômico também deu um certo trabalho, mas dessa vez eu tinha total certeza que daria certo. Segue abaixo as imagens dessa implementação.





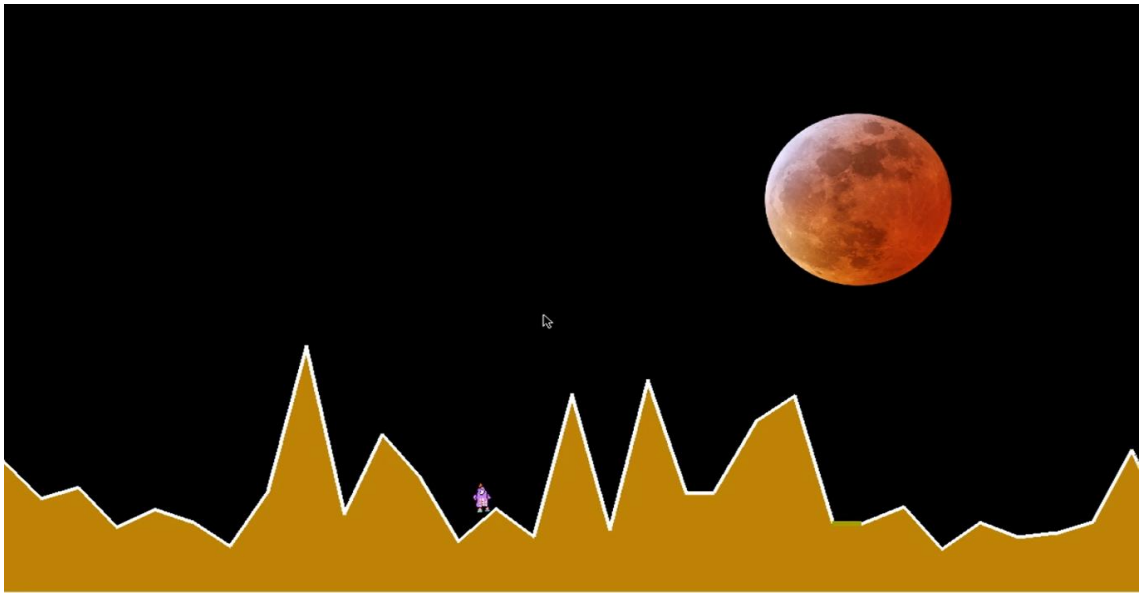


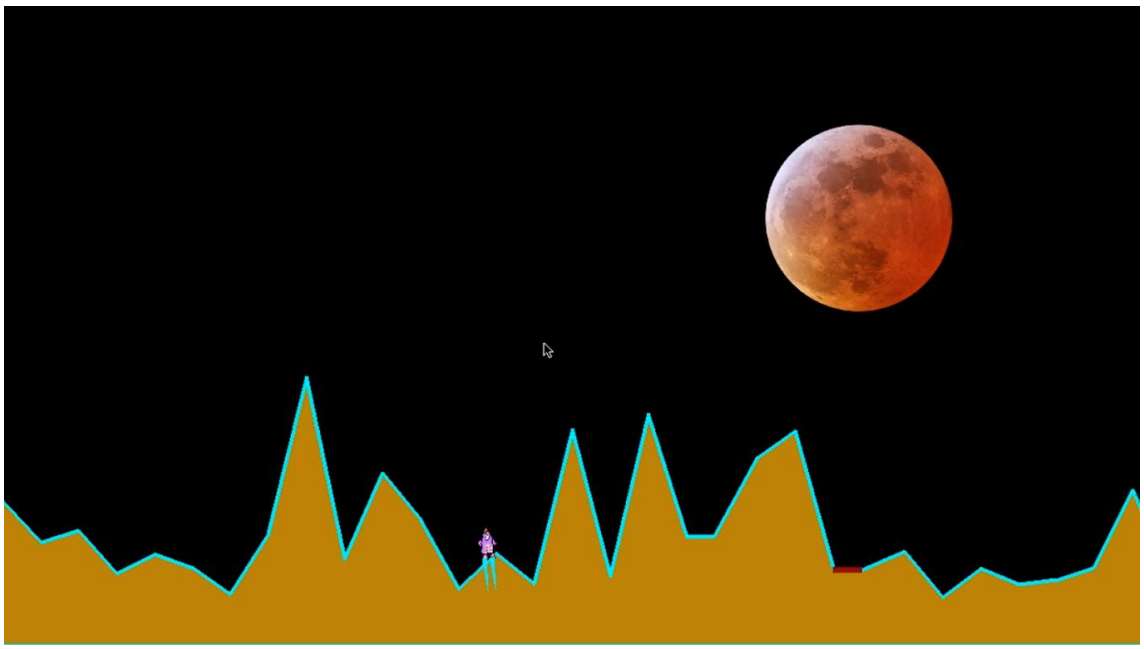




Neste caso eu também desenhei uma linha que percorria de vértice a vértice para poder obter assim como os retângulos, um feedback da localização desses novos pontos criados.

Por fim o sucesso de todo o esforço e dedicação para criar tanto um terreno dinâmico procedural quanto uma colisão que estivesse ao seu alcance, pude aproveitar essa linha como feedback de colisão para o jogador.





Tecnologias Envolvidas

Foi utilizado a IDE pycharme para a composicao

Cite e explique situações de compartilhamento, ajuda e cooperação entre grupos externos durante e fora do período de aula, e encontros remotos com o professor.

Única consulta de uma situação semelhante foi tirando dúvidas pelo Discord com o professor.

Você ou seu grupo utilizou grupos, sites e fóruns externos para ajudar no PI? Explique como. Diga se estes foram fornecidos pelo professor, colegas ou em sites de busca.

Utilizei sites de fóruns de programação, sites de aulas de geometria, dentro do projeto existe uma pasta com as referências.

Auto avaliação: Segundo seu ponto de vista, elabore um comparativo do que sabia antes e depois do projeto integrador.

Sem duvidas eu aprendi bastante, a parte conceitual e logica da estrutura por mim usada já vem na minha bagagem, mas com certeza o aprendizado com a biblioteca Pygame foi a maior adição nos meus conhecimentos. Também pude aperfeiçoar bastante meus conhecimentos em logicas de criação e manipulação de vértices no espaço, pois meu projeto foi desenvolvido em torno de 90% com polygon, lines, e rectangle. Usei pouquíssimos sprites por uma questão de auto desafio mesmo.