Universidade Estadual de Santa Cruz – UESC

Relatório da Implementação do Dado 3D para a Disciplina Computação Gráfica

Relatório de implementações realizadas por Iago Gomes Santana

Disciplina Computação Gráfica.

Curso Ciência da Computação

Semestre 2019.2

Professor Marcelo Ossamu Honda

Ilhéus – BA
2019

Linguagem Escolhida e justificativas

Para esse trabalho foi utilizada a linguagem Python, através da IDE PyCharm, onde os códigos foram escritos e compilados. A IDE fornece suporte para a linguagem e foi através de seu terminal que pude fazer a instalação do OpenGl (queria que o Windows também tivesse terminal decente :/) e ainda possui todas as biblioteca necessárias para a realização do trabalho.

O ponto forte de Python é a flexibilidade e abstração que a linguagem oferece. Conta ainda com uma vasta gama de bibliotecas disponíveis. E por fim, Python ainda tem uma grande vantagem pois reduz bastante a complexidade de algumas rotinas.

Em contrapartida, Python tem desvantagens quando o assunto é encontrar o erro feito pelo próprio programador, seja ele erro de lógica ou sintaxe. Isso se deve justamente por sua grande flexibilidade que permite fazer a mesma coisa de maneiras diferentes, como loops, condicionais, etc.

Como funciona

Logo de início, após rodar o código, o dado irá começar a girar e irá parar com alguma face aleatória para cima (pode acontecer de ser uma quina).

Para rodar o dado novamente (reroll) basta pressionar a tecla de espaço

Caso pressione a tecla enquanto o dado está rodando (começando ou parando de rodar), o dado irá reiniciar o processo de rolar

O cubo

Estratégia de Implementação:

Para fazer o cubo utilizei o código de referência disponibilizado pelo professor e baixei uma imagem da internet para usar como textura, dando assim uma forma mais apresentável ao meu dado.

Para fazer a textura aparecer somente nas faces do dado que eu queria, eu criei a função textura() que segue abaixo

```
def textura(filename):
    img = Image.open(filename)
    img_data = numpy.array(list(img.getdata()), numpy.int8)
    textureID = glGenTextures(1)
    glBindTexture(GL_TEXTURE_2D, textureID)
    glPixelStorei(GL_UNPACK_ALIGNMENT, 1)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST)
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL)
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, img.size[0], img.size[1], 0, GL_RGB, GL_UNSIGNED_BYTE, img_data)
    return textureID
```

E após isso eu precisava selecionar a parte da imagem que eu queria e recortar somente essa parte para um lado específico do meu dado. Para resolver isso tive de mapear a imagem usando o glTexCoord2f() antes de cada vertex. Nesse momento deve-se tomar muito cuidado com as coordenadas da imagem (que são sempre entre 0 e 1) porque o quadrado do recorte da imagem deve estar coincidindo com o quadrado da face do dado (basicamente é só não colocar coordenada no lugar errado).

Abaixo segue um exemplo de como foi feita uma das partes

```
85
86      # Frente 1
87      glTexCoord2f(0.29, 0.38)
88      glVertex3f(100.0, 100.0, 100.0)
89      glTexCoord2f(0.09, 0.38)
90      glVertex3f(-100.0, 100.0, 100.0)
91      glTexCoord2f(0.09, 0.14)
92      glVertex3f(-100.0, -100.0, 100.0)
93      glTexCoord2f(0.29, 0.14)
94      glVertex3f(100.0, -100.0, 100.0)
95
```

Obs.: vale lembrar que esse código não garante funcionamento para outras imagens

Dificuldades enfrentadas

Achar os números certos para serem usados como coordenadas. Pode ter sido por eu não ter usado uma ferramenta de visualização que possuísse régua, mas essa parte realmente foi chatinha de se implementar porque foi na base da tentativa e erro

Rotação

Estratégia de Implementação:

Como temos o glutMainLoop() que já deixa nosso código em looping infinito, não foi necessário nenhum looping, reaproveitei o do glut mesmo. A rotação foi implementada de forma que a cada vez que reiniciar o looping do glut, o dado sofra uma rotação (através do glRotated() que seja um grau (1°) maior que a anterior. A rotação deve ser feita assim pois sempre executamos o glLoadIdentity() na função showScreen(), logo temos de incrementar o ângulo da rotação sempre que looping for reiniciado.

Para controlar o ângulo da rotação eu uso as variáveis i; j; stop.

i: controla a angulação da rotação, sempre incrementando 1 e não tem limite de tamanho

j: também incrementa 1, mas ao se igualar ao valor de stop, a variável deve ser reiniciada pois ela serve para verificar se está na hora de parar de rodar

stop: variável que recebe um número aleatório múltiplo de 90 entre 540 e 2700. Essa variável serve para dizer que está na hora de parar o dado.

Também atribuí variáveis aleatórias para a,b e c que serão usados como argumentos referentes a x,y e z no glRotated(). Fiz isso porque assim ficaria mais imprevisível a rotação do dado.

Energia

Para dar energia ao dado e fazê-lo rodar mais devagar quando estivesse parando, eu usei a time.sleep() da lib time. Essa função dá um delay em segundos do valor que receber como argumento.

Então tudo que eu tive de fazer foi usar as variáveis que controlam a parada da rotação (j e stop) em uma equação matemática para ir aumentando o delay de acordo com o número de rotações, ficando assim ao fim:

```
time.sleep(1 / (stop - j + 1))
```

Obs.: precisa desse +1 no final para evitar a indeterminação matemática do caso de parada do if, quando j == stop

Reroll

Para implementar o reroll após teclarmos algo (“space”) no teclado, optei por usar uma função do glut e implementar mais duas.

A função glutKeyboardFunc() serve para pegarmos qual botão foi pressionado no teclado e retorna um valor do tipo byte

A função byte_to_int() serve para convertermos esse tipo byte em um int (que será usado posteriormente para comparação na tabela ascii

E por último, a função teclado() que recebe no formato byte o caractere pressionado pelo user, chama a byte_to_int() para converter esse valor em int e dps compara esse int com 32 (valor do “space” na tabela ascii). Caso seja verdadeiro são atribuídos novos números randômicos para a, b, c, j e stop, o que faz com que se reinicie a rotação do dado