Gerando um Braço Mecânico Virtual com PyOpenGL

Daniel T. B. Torres, **William A. M. de Morais**

Ciência da Computação – Universidade Estadual do Rio Grande do Norte (UERN)
Campus Avançado de Natal - CaN
Av. Dr. João Medeiros Filho, nº3419 – 59120-555 – Natal – RN – Brasil

danielteodolino@gmail.com, will.mendes37@gmail.com

Resumo

Este relatório tem o objetivo de apresentar o processo de modelagem de um braço mecânico virtual com a utilização da *Application Programming Interface* (API) Python *Open Graphics Library* (PyOpenGL). A proposta dessa API é simplificar a síntese de imagens virtuais. Sendo assim, neste trabalho, com o auxílio da API PyOpenGL, foi gerado um braço mecânico, e este, será controlado por meio do teclado do computador, utilizando teclas específicas. Para isso, inicialmente, foi realizado um levantamento bibliográfico de histórico e conceitos dessa API, com a finalidade de compreender algumas das funções que foram aplicadas no desenvolvimento do código. Como resultado obtido, foi possível desenvolver formas geométricas que simulam um braço mecânico virtual.

Palavras-chaves: OpenGL, API, Síntese de Imagens.

1. Introdução

Python Open Graphical Library (PyOpenGL) pode ser definida como uma interface de software (API – *Aplication Program Interface*) para aceleração da programação de dispositivos gráficos, com aproximadamente 120 comandos para especificação de objetos e operações necessárias para a produção de aplicações gráficas interativas 3D. Na verdade, é possível classificá-la como uma biblioteca de rotinas gráficas para modelagem 2D ou 3D, extremamente portável e rápida, possibilitando a criação de gráficos 3D com excelente qualidade visual e rapidez, uma vez que usa algoritmos bem desenvolvidos e otimizados pela Silicon Graphics. Justamente pela sua portabilidade, não possui funções para gerenciamento de janelas, interação com o usuário

ou arquivos de entrada/saída. Cada ambiente, como, por exemplo, o Microsoft Windows, possui suas próprias funções para esses propósitos.(Azevedo, 2003).

Como descrito, PyOpenGL não é uma linguagem de computação como C, C++ ou Python, é uma API ou biblioteca para desenvolvimento de aplicações gráficas 3D renderizadas em tempo real.

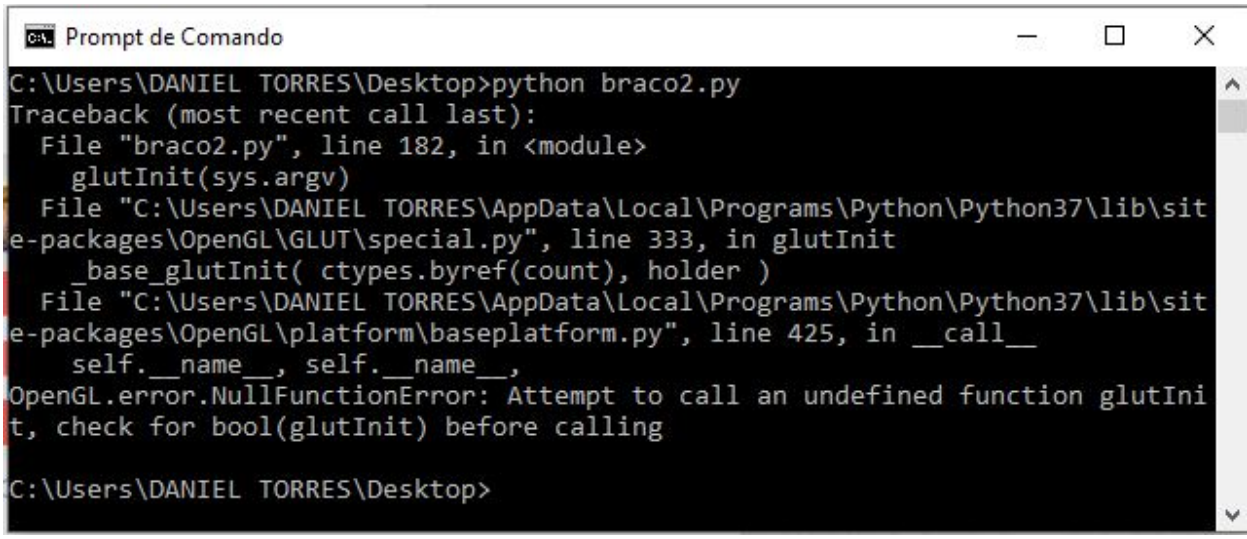
2. Modelagem do Braço Mecânico

Para modelar o braço mecânico, foi necessário instalar alguns o pacote PyOpenGL e a biblioteca OpenGL Utility Toolkit (GLUT).

A biblioteca GLUT disponibiliza funcionalidades para PyOpenGL e o principal objetivo é a abstração do sistema operacional fazendo com que os aplicativos sejam multiplataforma. A biblioteca possui funcionalidades para criação e controle de janelas, e também tratamento de eventos de dispositivos de entrada (mouse e teclado). Também existem rotinas para o desenho de formas tridimensionais pré-definidas (como cubo, esfera, bule, etc).

Neste momento, vale ressaltar as dificuldades e a forma que elas foram contornadas

Em um primeiro momento, foram realizadas inúmeras tentativas de manipulação e modelagem utilizando a API e a biblioteca no Sistema Operacional (SO) Windows 10, porém, existem conflitos que ainda não foram identificados. A API e a biblioteca estão instaladas no SO, entretanto, ao executar o código fonte é apresentado erros que indicam a ausência de módulos da GLUT.



Para contornar essa dificuldade, foi realizado a instalação da API e biblioteca, em uma máquina Linux Ubuntu e outra com a distribuição Debian 10. Inicialmente atualizamos o Python para a versão 3.7, atualizamos o PIP e para instalar os pacotes, utilizamos os seguintes comandos:

1) Instalaçao do PyOpenGL:

```
$ pip install PyOpenGL
```

2) Instalação da Biblioteca GLUT

```
$ pip install PyOpenGL PyOpenGL_accelerate
```

3. Codigo

Iniciamos o código importando as bibliotecas necessárias:

```
1  import OpenGL
2  from OpenGL.GLUT import *
3  from OpenGL.GL import *
4  from sys import argv
5  from random import uniform
6  from OpenGL.raw.GLU import gluPerspective
```

Em seguida, definimos as variáveis globais e inicialimos elas com valor zero. Essas variáveis servirão para controlar a movimentação dos objetos dentro da cena.

```
8   cotovelo = 0
9   mao = 0
10  dedos = 0
11  polegar = 0
12  indicador = 0
13  medio = 0
14  horizontal = 0
```

Dando continuidade, temos a def display (). Aqui serão implementadas todas as formas geométricas, suas posições dentro da cena, suas respectivas cores, formatos e tudo que for relacionado a características “físicas”.

Aqui serão utilizadas com grande frequência as definições da biblioteca GLUT e elas podem ser identificadas facilmente dentro do código, pois inicial com a sigla gl.

Para a modelagem do braço, foi feito um SolidCube para representar o Braço e Antebraço e SolidSphere para representar o cotovelo.

```
16  def display():
17      global cotovelo, mao, dedos, polegar, indicador, medio, horizontal
18
19      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
20
21      glPushMatrix()
22      glTranslatef (0.0, 2.6, 0.0)
23      glPopMatrix()
24
25      glPushMatrix()
26      glTranslatef (horizontal, 1.5, 0.0)
27
28      ## Desenho do braco parte superior
29      glPushMatrix()
30      glColor3f(0.0, 0.0, 255.0) #cor do braco_superior. Funcao recebe 3 parametros (RGB) do tipo float
31      glScalef (0.4, 2.0, 0.4) #glScalef(ex, ey, ez) que altera a escala do objeto ao logo dos eixos coord
32      glutSolidCube(1.0) #glutSolidCube renderiza um cubo solido
33      glPopMatrix()
34
35      ## Desenho do cotevelo para juncao da parte superior e inferior
36      glTranslatef (0.0, -1.0, 0.0)
37      glRotatef (cotovelo, 0.0, 0.0, 1.0)
38      glColor3f(0.0, 0.0, 255.0)
39      glutSolidSphere(0.30, 50, 50) # glutSolidSphere(GLdouble radius, GLdouble slices, GLdouble stack) re
40      glTranslatef (0.0, -1.0, 0.0)
41
```

Em seguida, temos a def teclas. Essa função recebe como parâmetros uma tecla e configura a forma geométrica de acordo com essa tecla e o posicionamento dos demais elementos dentro da cena.

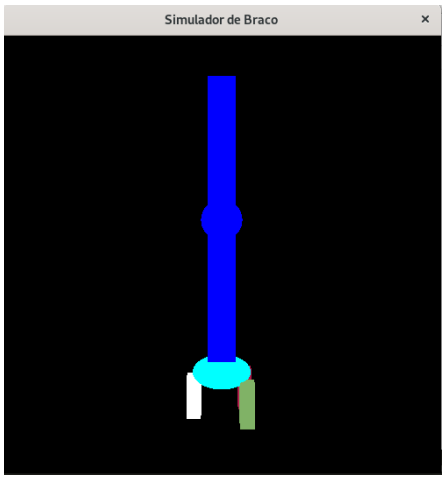
```
112    def teclas (key, x, y):
113        global cotovelo, mao, dedos, polegar, indicador, medio, horizontal
114        if key == 'd':
115            if horizontal < 2.5:
116                horizontal = horizontal + 0.1
117            glutPostRedisplay()
118        elif key == 'e':
119            if horizontal > -2.5:
120                horizontal = horizontal - 0.1
121            glutPostRedisplay()
122        elif key == 'r':
123            if cotovelo != 110:
124                cotovelo = (cotovelo + 5) % 360
125            glutPostRedisplay()
126        elif key == 'R':
127            if cotovelo != 250:
128                cotovelo = (cotovelo - 5) % 360
129            glutPostRedisplay()
130        elif key == 'H':
131            mao = (mao + 5) % 360
132            glutPostRedisplay()
133        elif key == 'h':
134            mao = (mao - 5) % 360
135            glutPostRedisplay()
136        elif key == 'T':
137            if dedos != 30:
138                dedos = (dedos + 5) % 360
139            polegar = 0
```

Por ultimo, o main reúne todas as funções e as colocam em um loop.

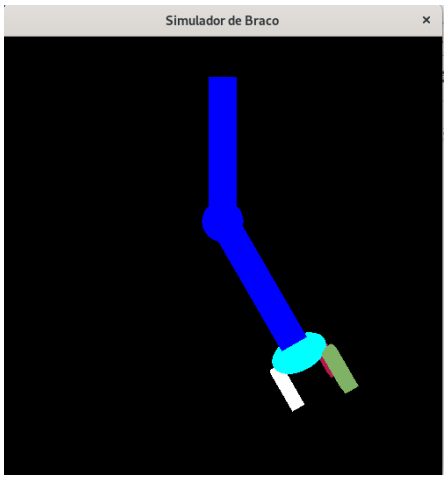
```
181    if __name__ == '__main__':
182        glutInit(sys.argv)
183        glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH)
184        glutInitWindowSize(500, 500)
185        glutInitWindowPosition(100, 100)
186        #glutFullScreen()
187        glutCreateWindow("Simulador de Braco")
188        glEnable(GL_DEPTH_TEST)
189        glutDisplayFunc(display)
190        glutIdleFunc(display)
191        glutReshapeFunc(modelagem)
192        glutKeyboardFunc(teclas)
193        glutMainLoop()
```

4. Resultados Obtidos.

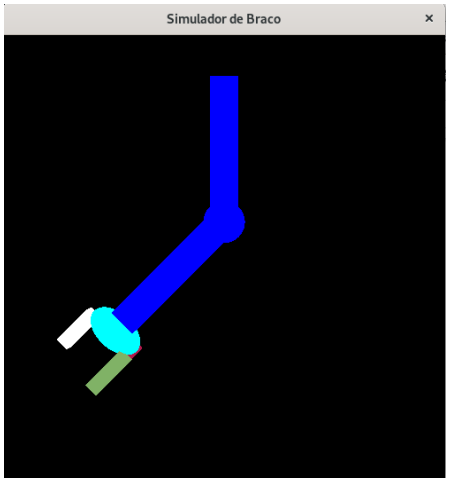
Posição Inicial



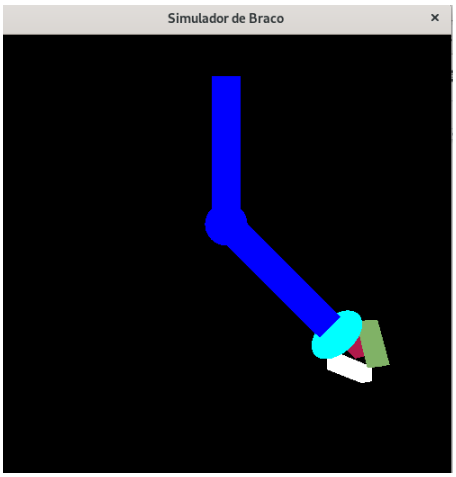
Posição após r ser pressionado



Posição após R ser pressionado



Posição após p, i, m, h serem pressionados



5. Referências

AZEVEDO, E. CONCI, A. **Computação Gráfica** - Geração de imagem.São Paulo: GEN LTC, 2003.

Documentação GLUT <https://www.opengl.org/resources/libraries/glut/>. Acesso em 03/03/2020.

PyOpenGL < http://pyopengl.sourceforge.net/> Acesso em 03/03/2020.

KILGARD, M. **GLUT APIversion 3.** 1996. Disponível em:< https://www.opengl.org/resources/libraries/glut/spec3/node113.html >. Acesso em: 03 mar. 2020