Tarea 3: Endless Runner

Francisco Muñoz P.

Profesora: Mara Cecilia Rivara Z.

Universidad de Chile

21 de noviembre de 2017

Descripción

Objetivos.

Carracterísticas Básicas

- Personaje simple
- Escena de figuras 3D
- Interaccines con escenario

Adicionales

- Escena rota
- Puntaje
- Menus
- Modos de juego
- Musica
- utilizar git

Paralelepipedors

Mezcla de GL_Triangles, dos trangulos arman cuadrilatero, 6 cuadrilatero =¿ paralelepipedo

```
def master_parallelepiped(self, number):
    idx_ver = [[-1, -1, 1], [-1, 1, 1], [-1, -1, -1], [-1, 1, -1],
               [1, -1, 1], [1, 1, 1], [1, -1, -1], [1, 1, -1]]

    idx_faces = [[1, 0, 2, 3], [5, 4, 6, 7], [4, 0, 2, 6], [5, 1, 3, 7],
                 [3, 7, 6, 2], [1, 5, 4, 0]]

    # la idea es generar el paralelepipedo maestro segun set_displacement_pos,
    # y luego agregar la posicion maestra.

    for face in idx_faces:
        v1, v2, v3, v4 = self.individual_rect(idx_ver, face, number)
        cuadrilatero(v1, v2, v3, v4)
```

tunnel

agrupando varios paralelepipedos en una estructura de octaedro se genera un tunel. Colores de los paralelepipedos varian

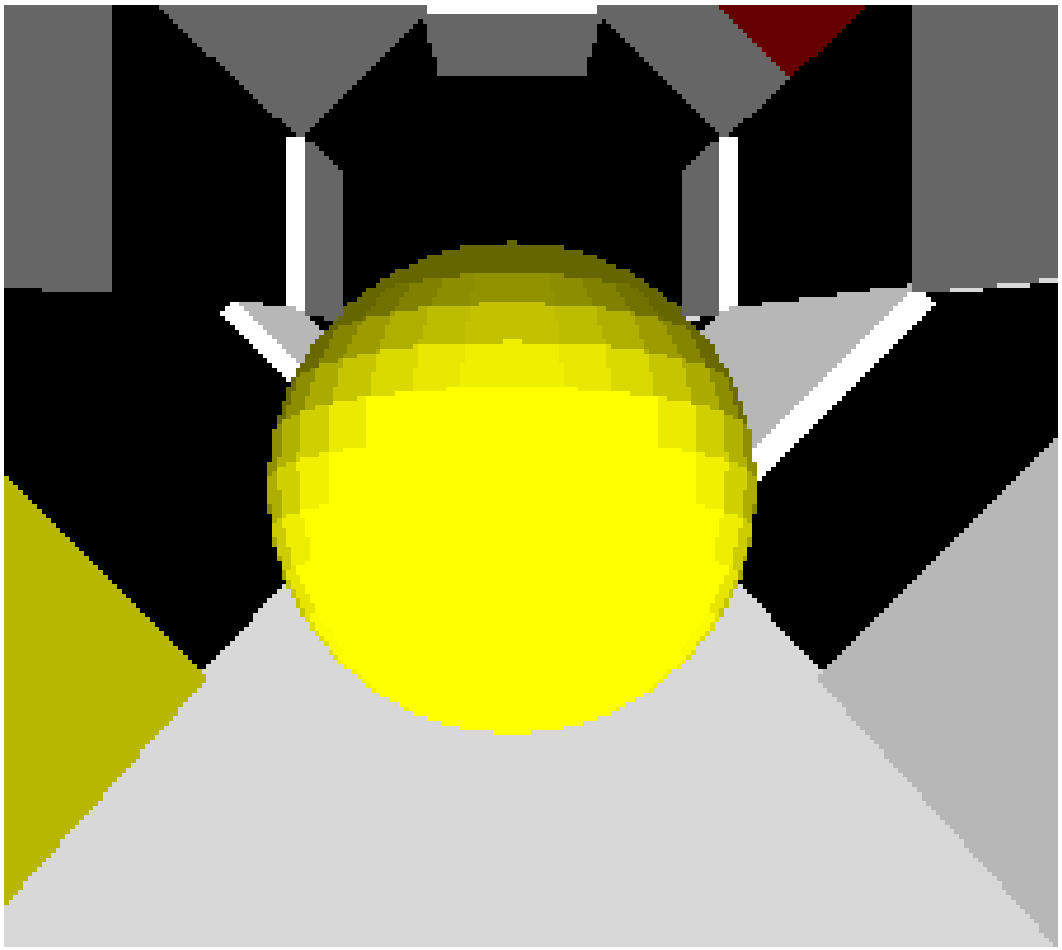
```
def crear(self):
    """
    crea cada uno de los 8 paralelepipedos en su correspondiente posicion,
    trasladando un paralelepipedo maestro, usa atributo type para definir
    el tipo de paralelepipedo a crear, puede ser  nulo(0), simple(1)
    o comodin (2)
    :return:
    """
    self.lista = glGenLists(1)
    glNewList(self.lista, GL_COMPILE)
    glEnable(GL_COLOR_MATERIAL)
    glBegin(GL_TRIANGLES)

    for i in range(8):
        if self.tipos[i] == 1 or self.tipos[i] == 2:  # vacio
            glColor3f(0.0, 0.1, 0.0)
            continue
        elif self.tipos[i] == 0:  # blanco
            glColor3f(1.0, 1.0, 1.0)
        elif self.tipos[i] == 3:  # amarillos, otrogan mas tiempo de vida
            glColor3f(1.0, 1.0, 0.0)
        elif self.tipos[i] == 4:  # rojos, restan tiempo de vida
            glColor3f(1.0, 0.0, 0.0)
        else: # lo demas es vacio
            continue
        self.master_parallelepiped(i)

    glEnd()
    glEndList()
```

personaje

Personaje generado simple, consta de una sola esfera



Salto lateral

Personaje salta hacia los lados usando teclas A o B, esto rota todo el escenario

```
def spin(self):
    if self.left and not self.right:
        ang = -self.discrete_rotation()
    elif self.right and not self.left:
        ang = self.discrete_rotation()

    if all(self.spin_conditions()):
        # print("modifico la rotacion segun Delta_ang: ",
        # print("instant = ", self.instant)
        self.upgrade_vertica_pos("small_jump")
        Stage.modify_fi(self, ang)
        self.instant += 1

    if self.instant >= int(self.instants / 2) + 1 \
            and not self.jumping and not self.falling:
        self.instant = 1
        # Stage.modify_fi(self, 45)
        # print("nuevo master fi = ", self.ang)
        self.spinning = False
        self.right = False
        self.left = False
```

Salto vertical

Personaje salta hacia delante usando tecla SPACE.

```
def jump(self):
    if all(self.jump_conditions()):
        self.upgrade_vertica_pos("jump")
        self.instant += 1

    if self.instant >= self.instants \
            and not self.falling and not self.spinning:
        self.instant = 1
        self.jumping = False
```

Caer

Personaje puede caer cuando se encuentra un un sector sin "piso".

```
def fall(self):
    oct = self.octagons[1]
    run = True
    if all(self.fall_condition(oct)):  # cumple todas las condiciones de caida
        self.upgrade_vertica_pos("fall")
        self.instant += 1
        self.falling = True  # esto permitira bloquear otras acciones

    if self.instant >= self.instants \
            and not self.jumping \
            and not self.spinning:  # luego de cierto tiempo "muere"
        self.falling = False
        self.muerte = True
        run = False
        # aqui debemos generar la muerte

    return run
```

gana tiempo

El juego consta de terminal escenario antes que se acabe tiempo.
Paralelepipedos amarillos dan bonus de tiempo.

```
def add_time(self):
    oct = self.octagons[1]
    get_more_time = False
    lower_type = self.get_type_lower_block(oct)
    if all(self.time_conditions(oct)) and lower_type == 3:
        get_more_time = True

    return get_more_time
```

pierde tiempo

Tambien, los bloques rojos quitan tiempo. Si se acaba el tiempo del jugador, este pierde.

```
def rest_time(self):
    oct = self.octagons[1]
    lower_type = self.get_type_lower_block(oct)
    get_minus_time = False
    if all(self.time_conditions(oct)) and lower_type == 4:
        get_minus_time = True

    return get_minus_time
```

Texto

Se agrega texto en el juego de dos formas, utilizando GLUTBITMAP y utilizando pygame.font

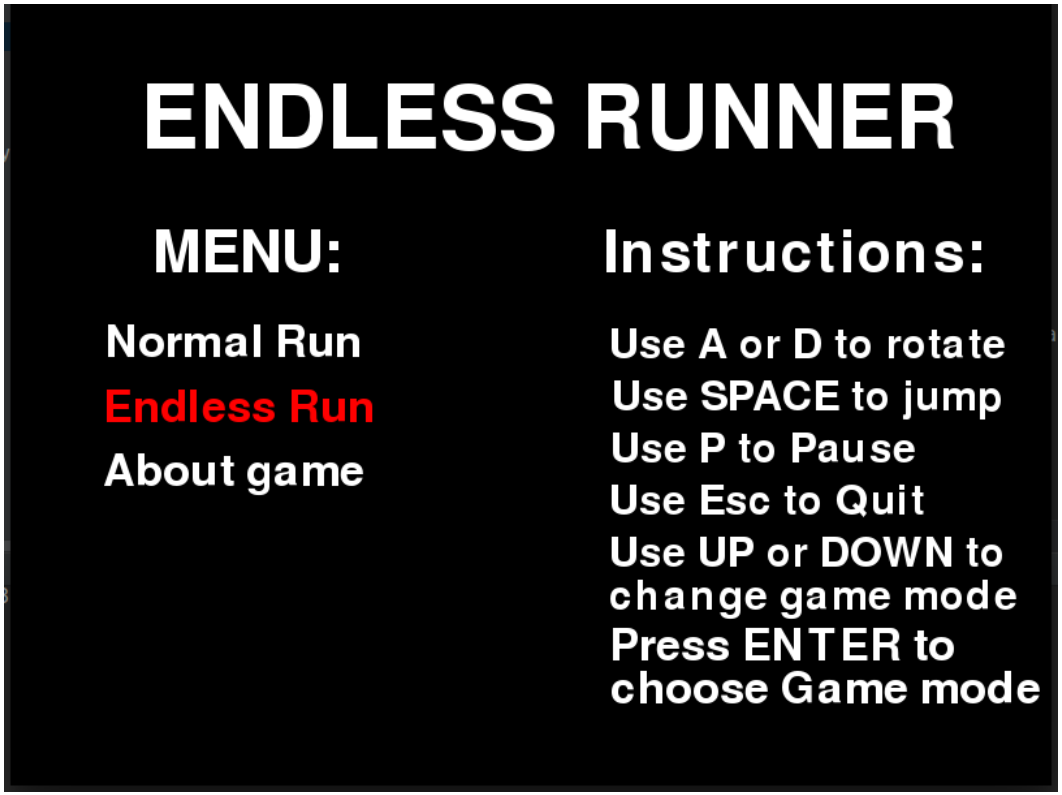
```
def drawTextwithglut(value, x, y):

    glRasterPos2i(x, y)
    lines = 0
    ##  import pdb
    ##  pdb.set_trace()
    for character in value:
        if character == '\n':
            glRasterPos2i(x, y - (lines * 18))
        else:
            glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, ord(character))

def draw_text_box(pos, width, text, color=(255, 255, 255, 0),
                  fondo=(255, 0, 0, 0)):
    tamanno = int(width / 2)
    font = pygame.font.Font(None, tamanno)
    text_surface = font.render(text, True, color, fondo)
    text_width = text_surface.get_width()
    text_height = text_surface.get_height()
    x = pos[0] + (width - text_width) / 2
    y = pos[1] + (width - text_height) / 2
    position = (x, y)
    text_data = pygame.image.tostring(text_surface, "RGBA", True)
    glRasterPos2d(*position)
    glDrawPixels(text_surface.get_width(), text_surface.get_height(),
                 GL_RGBA, GL_UNSIGNED_BYTE, text_data)
```

Menus

Se generan Menus interactivos para iniciar juego, luego de morir y luego de ganar.



Menus

Se generan Menus interactivos para iniciar juego, luego de morir y luego de ganar.

Escenario

Se genera un escenario predeterminado para el modo normal y otro de forma aleatoria para el modo sin fin

```
def randon_types(self):
    """
    genera un arreglo de los tipos de paralelepipedos a crear
    :return:
    """
    if not self.pregenerated: # no hay pregenerado
        tipos = [int(i) for i in np.random.exponential(1, size=8)]
    else: # si hay pregenerado
        if self.preg_counter < len(map1): #toma los tipos del mapa
            tipos = map1[self.preg_counter] # aumenta el contador
        else:
            tipos = [0, 0, 0, 0, 0, 0, 0, 0] # despues agrega escena plana
        self.preg_counter += 1

    return tipos
```

Camara

La camara se deja fija y se mueve el mundo, el personaje no se desplaza por el escenario. La luz se genera de forma realista sobre el escenario

```
# camara
glLoadIdentity()
gluLookAt(camPos.x, camPos.y, camPos.z,  # posicion
          camAt.x, camAt.y, camAt.z,  # mirando hacia
          0.0, 0.0, 1.0)  # inclinacion
# luz
glLightfv(light, GL_POSITION, l_position)
glLightfv(light, GL_AMBIENT, l_ambient)
glLightfv(light, GL_SPECULAR, l_specular)
glLightfv(light, GL_DIFFUSE, l_diffuse)
glLightf(light, GL_CONSTANT_ATTENUATION, l_constant_attenuation)
glLightf(light, GL_LINEAR_ATTENUATION, l_linear_attenuation)
glLightf(light, GL_QUADRATIC_ATTENUATION, l_quadratic_attenuation)
glLightf(light, GL_SPOT_CUTOFF, l_spot_cutoff)
glLightfv(light, GL_SPOT_DIRECTION, l_spot_direction.cartesianas())
glLightf(light, GL_SPOT_EXPONENT, l_spot_exponent)

glEnable(light)

pygame.display.flip()  # cambiar imagen
clock.tick(fps)  # esperar 1/fps segundos
```

Musica

Se agrega musica de fondo y efectos de sonido utilizando pygame.

```
pg.mixer.init(11025)
pg.mixer.music.load("sounds/Electrical-of-cosmic.mp3")
pg.mixer.music.set_volume(0.4)
pg.mixer.music.play(-1)
menu1 = pg.mixer.Sound("sounds/sfx_menu_move3.wav")
menu2 = pg.mixer.Sound("sounds/sfx_menu_select2.wav")
long_jump = pg.mixer.Sound("sounds/sfx_movement_jump16.wav")
short_jump = pg.mixer.Sound("sounds/sfx_movement_jump8.wav")
die = pg.mixer.Sound("sounds/sfx_sounds_falling7.wav")
win = pg.mixer.Sound("sounds/Ta-Da.wav")
pause = pg.mixer.Sound("sounds/sfx_sounds_pause1_in.wav")
more_time = pg.mixer.Sound("sounds/sfx_movement_portal1.wav")
less_time = pg.mixer.Sound("sounds/sfx_movement_portal3.wav")
sounds = [menu1, menu2, long_jump, short_jump, win, pause,
          more_time, less_time]
```

Git

Durante todo el desarrollo se utilizo GitHub para registar los avances, los comandos basicos utilizados son:

- git clone: permite clonar un repositorio only
- git status: revisa el estado del repositorio only respecto al locar para ver inconsistencias
- git add: usado para agregar nuevos archivos o cambios generados al repositorio.
- git commit: usado para comentar los cambios agregados, siempre es necesario hacer commit
- git push: empuja los cambios generados al repositorio online.

Resultados

Dificultades

- Complejidad en la creacion de escenas 3D
- Gran volumen de codigos necesario para funcionamiento
- Interacciones complejas de modelar
- Abuso de condicionales, en especial, en la interacciones con las teclas del teclado.

Resultados

Conclusiones

- El potencial de utilizar OpenGl + pygame es mucho mas notorio al crear escenas 3D
- Engorroso de programar pero con gran versatilidad
- Juego entretenido :)
- Interesante(y laaaaaarga) tarea

Juego

Disfrutad del juego!