Problema 1

Solución Problema 1

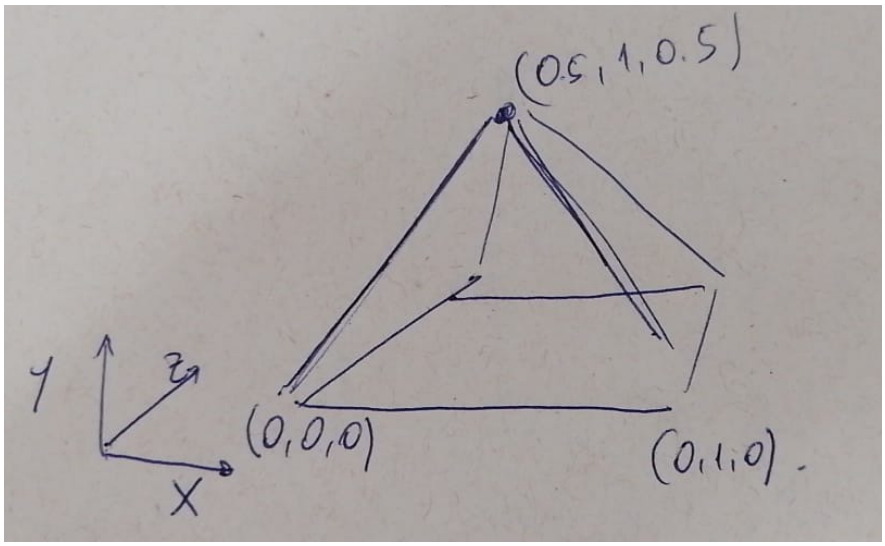
1. Este problema posee varias formas de ser abordado, la solución representa una de ellas.

```
controlador:
    posX = 0

apretarTecla:
    if left:
        posX -= dt
    elif right:
        posX += dt

nubes = rectangulo(nubes, 30, (0, 0, -3), (0, 0, 1), 30)
montanas = rectangulo(montana, 30, (0,0,-2), (0,0,1), 30)
bosque = rectangulo(bosque, 30, (0, 0, -1), (0, 0, 1), 30)
camino = rectangulo(camino, 30, (0, 0, 0), (0, 0, 1), 30)

near = top = right = 1
far = bottom = left = -1
proyeccion(M) = frustum(near, far, top, bottom, right, left)
while True:
    apretarTecla()
    lookAt((controlador.posX, 0, 1), (0, 1, 0), (controlador.posX, 0, -1)
    dibujar(objeto, camara, proyeccion(M)) #objeto corresponde a todos
                                           # los rectangulos
```

   Notar que para tener el efecto de parallax, la posición de la cámara y el punto de observación tiene que moverse coordinadamente, además de que se necesita una proyección en perspectiva, por lo que se dibujan los rectángulos (lo suficientemente grandes) y se colocan en distintas posiciones del eje Z.
   Luego, se define la matriz de proyección (usando los parámetros top, bottom, near, far, left y right) y en el ciclo se modifican los valores de la matriz de vista en función de la posición del controlado, dibujando todos los rectángulos.

2. Para el problema también se tienen distintas soluciones dependiendo de los valores asignados a la cúspide de la pirámide y de los ejes de referencia, ésta corresponde a una elección: (se usará el sistema que se representa en el dibujo a continuación)

Control 1



Vértices de la pirámide

```
v   0    0   0
v   0    1   0
v   0    1   1
v   0    0   1
v   0.5  1   0.5
```

Coordenadas de textura:

```
vt  0    1
vt  1    1
vt  1    0
vt  0    0
vt  0.5  0.5
```

Caras:

```
f  1/1  2/2  5/5
f  2/2  3/3  5/5
f  3/3  4/4  5/5
f  4/4  1/1  5/5
```

Problema 2

Solución Problema 2

- Lanzar desde un plano determinado rayos hacia la figura. Estos rayos se separan una distancia ΔX y ΔY de cada uno. Si el rayo atraviesa la figura, suman $\Delta X \cdot \Delta Y \cdot \Delta Z_{ij}$. Donde ΔZ_{ij} es la distancia comprendida entre el punto en que el rayo entra en la figura y sale de ella. Si el rayo no atraviesa la figura, no se suma esto al volumen.

  De esta manera, el volumen total será:

  $$\sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} \cdot \Delta Z_{ij}.$$

  Donde $A_{ij} = \Delta X \cdot \Delta Y$. Pero considerando un grillado asignando índices i y j para cada celda.

- El modelo Flat es muy barato computacionalmente. Asigna un color único a cada triángulo que compone la figura. Esto hace que se vea poco realista como en juegos de 8 bits.
  El modelo Gourad es intermedio en costo computacional. Asigna un color a cada vértice según la luz y las normales, luego interpola estos valores en los puntos intermedios para obtener el color en cada pixel. Esto logra un mejor efecto en superficies suaves.
  El modelo Phong es caro computacionalmente. Pero logra gran realismo. Calcula el color final de la figura en cada pixel o fragmento (hasta aquí debería tener todo el puntaje). Así que este costo es mucho mayor para figuras de caras grandes de forma casi independiente de su cantidad de vértices.

- A partir del enunciado se extraen los siguientes datos:

	x	y	z	
v =	(0.8,	-10,	0.5)	# viewPos
s =	(0.5,	0.25,	0.5)	# FragPos
l =	(0.5,	-20,	0.5)	# LightPos
K_a =	(0.2,	0.2,	0.2)	
K_d =	(0,	1,	0)	
K_s =	(1,	1,	1)	
L_a =	(0.1,	0.1,	0.1)	
L_d =	(1,	1,	1)	
L_s =	(1,	1,	1)	

Notar que el punto S pedido se encuentra exactamente en el centro entre P, Q y R. ¿Cómo será la normal? Recordemos que se interpola a partir de los vértices. Entonces: $n_s = (0{,}5, -0{,}5, 0{,}5)$.

Ahora hay que considerar el color final según la fórmula del color final. Esta es igual para los tres modelos. Sin embargo, el punto que se considera difiere entre los 3.
Para el caso de Phong, será:

$I_{phong} = K_a L_a + \frac{1}{k_c + k_l d + k_q d^2}(K_d L_d (l \cdot n_s) + K_s \cdot L_s (v \cdot r)^\alpha)$

Para el caso de Gouraud, será:

$I_{Gouraud} = \frac{1}{3}\sum K_a L_a + \frac{1}{k_c + k_l d + k_q d^2}(K_d L_d (l \cdot n_i) + K_s \cdot L_s (v \cdot r_i)^\alpha)$

Donde cada índice i corresponde a un vértice que conforma la pirámide. Se agregó el tercio porque es una interpolación en el punto medio de los tres vértices. Las normales de los vértices se interpolan pues no hay información de caras vecinas.

En el caso de Flat simplemente se elige un vértice y el color resultante será ese. Mientras se haya elegido algún vértice de los indicados, el resultado está correcto.

Problema 3

Solución Problema 3

1. Tanto la parte 1 como la 2 son dependientes de la creatividad del estudiante, en el caso de la pregunta 1 se adjuntarán dos curvas y sus definiciones mediante cadenas de letras (F, L, R). En el primer caso se tiene un estimado de la dimensión fractal (1.934007183)

Nombre	Regla	Ángulo
Levy	F → RFLFFLFR	90°
Koch adaptado	F → FLFRRFLF	90°

   y en la curva de Koch el factor de escala es 3.

2. Para esta parte se propone el siguiente algoritmo (todo depende de como se use la función random para dibujar.
   Recordando que la recla para la creación de un árbol fractal según la lectura es $F \rightarrow F[RF]F[LF]F$:

```
arbol_etapa(P, Q,n):
    if n == 0:
        hoja(x, n)
    else:
        P1 = P + (Q - P) * random()
        P2 = P + (Q - p) * random()
        linea(P, Q, n)
        LargoLinea = ||Q-P||
        AnguloRama = pi/2 - pi/6
        rama_x = (LargoLinea * random()) * sen(AnguloRama)
        rama_y = (LargoLinea * random()) * cos(AnguloRama)
        arbol_etapa(P1, P1 + (rama_x, rama_y), n-1)
        arbol_etapa(P2, P2 + (-rama_x, rama_y), n-1)
        arbol_etapa(P, P1, n-1)
        arbol_etapa(P1, P2, n-1)
        arbol_etapa(P2, Q, n-1)

arbol(x, n):
    arbol_etapa(x, x + (random(), x_y), n-1)
```

(Los ángulos de las ramas probablemente no están bien calibrados en el algoritmo, lo importante es la idea del como construirlo)
Cabe destacar que para esta solución se crea una función auxiliar que crea en la misma iteración un punto para dibujar un tronco/rama, notando que después para una etapa cualquiera, si se tiene que es la iteración final (n=0), entonces se dibuja una hoja. En caso contrario, se dibuja la linea entre x e y, se buscan dos puntos aleatorios en el tronco (x1 y x2), se calcula el tamaño del tronco para obtener el tamaño de las ramas nuevas (para este ejemplo, ambas ramas tendrán igual tamaño, pero se pueden aleatorizar por separado, para un mejor resultado). Una vez calculado las posiciones del final de la rama (rama_x en el eje x y rama_y en el eje y), se dibujan entonces las nuevas ramas en una etapa menos. (Notar que como en la lectura, se realiza la etapa para cada F, entonces se realiza la iteración sobre todo el tronco del árbol)

3. Las dos características principales de los fractales son:

   - Autosemejanza: Una porción puede ser vista como una réplica del todo, en una escala menor.
   - Independencia de la escala: Como la forma de construir la curva es recursiva y autosemejante, entonces las curvas fractales son independientes de las escalas a las que son vistas

   De esta manera, las curvas fractales no se pueden describir por la geometría euclidiana. La dimensión fractal es una métrica que indica qué tanta porción del espacio ocupa el

fractal, o visto de otra manera, cuántos puntos en ese espacio logra incorporar en la curva. Esta unidad se calcula usando la cantidad de objetos que se van creando en cada división y del factor de escala.