Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación
CC3501-1 Modelación y Computación Gráfica para Ingenieros

Tarea 3B

Mr. Pedro's hotel

Alumna: Julia Paredes

Profesor: Daniel Calderón

Auxiliares: Alonso Utreras - Nelson Marambio

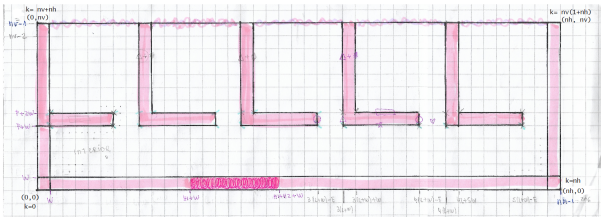
Fecha de entrega: 14 de Julio, 2020

Solución propuesta

Lo primero que hice fue definir la discretización del hotel. Del enunciado de la tarea entendí que los muros deben estar presentes en el dominio, pues ellos tienen condición de Neumann nula. La discretización fue rectangular con un $h = 0{,}1$, Como se trata de resolver la ecuación de Laplace queda el stencil de 5 puntos:

$$\phi_{i-1,j} + \phi_{i+1,j} + \phi_{i,j-1} + \phi_{i,j+1} - 4\phi_{i,j} = 0 \tag{1}$$

El cual queda de esa forma para todos los puntos del hotel que no sean muros. Para las orillas de los muros, de su condición Neumann nula se cumple que estos puntos tienen la misma temperatura que el punto (perteneciente a la habitación o pasillo) más cercano a ellos, en el caso de las esquinas de los muros, su temperatura queda como un promedio entre las temperaturas de los puntos vertical y horizontal pertenecientes a la habitación o pasillo más cercanos a ellos. Y para los puntos dentro del muro les otorgué la temperatura de uno de los extremos del muro. En cuanto a los puntos ubicados en las ventanas, estos cumplen sus condiciones respectivas, ya sean Neumann igual a *window_loss* cuando se encuentran cerradas, o Dirichlet igual a *ambient_temperature* cuando están abiertas. En el código cada parte de la discretización tiene comentada una palabra (circulo, estrella, corazón, sombreado), la cual corresponde al dibujo de discretización:



Todo esto se realizó bajo la función $finnite\_differences()$, la cual no recibe parámetros, pues funciona con los valores ya entregados al momento de ejecutar el programa $hotel - solver.py$. Esta función retorna y grafica la matriz de solución de las temperaturas, matriz que es guardada con el nombre asignado por el usuario en "$filename$". En $hotel - solver.py$ también se calcula el gradiente en x e y para todos los puntos que no sean pertencientes a los muros con la función $calculate\_gradient\_foward(V)$ la cual recibe como parametro la matriz $filename$, de esta manera no se crean los gradientes muy grandes en la diferencia de temperaturas entre las piezas, ya que recordemos que los muros tienen ambas temperaturas. Se da la opción de graficar esta matriz descomentando la última linea del programa. La matrices de gradientes en x e y son guardadas para su próxima utilización.

En el siguiente programa, $hotel-viewer.py$, lo primero que se realizó fue el hotel con la función $createhotel()$ la cual fue implementada con grafos de escena, destaco que dibujé el calefactor de color café en la pared.

En cuanto a la cámara, esta está programada de manera que a donde miro depende del ángulo que se produce al presionar la tecla derecha o izquierda, y me ubico en el punto que se produce al presionar las teclas arriba y abajo. Esto fue calculado con coord. cilíndricas. El piso del hotel se creó con la función $createSuelo(ancho, largo, funciondegradé, matrizfilename)$ ubicada en $basic\_shapes$, la cual recibe como parámetros el ancho y largo de la matriz $filename$, una función que se encarga de asignarle el color adecuado a cierta temperatura y la matriz $filename$. Esta ecuación para cada punto de la discretización genera un cuadrado con los puntos siguientes en x e y.

Las curvas de nivel se crearon con una matriz de "$voxeles$", la cual es creada con la función $my\_marching\_cube(filename, lista)$ la cual recibe la matriz $filename$ y una lista con 10 temperaturas entre la mínima y máxima. Pero esta matriz de volexes en cada punto que cumple con alguno de los valores

de temperatura guarda el índice + 1 de la temperatura que posee de la lista entregada de temperaturas, de esta forma en la función $funcioncurvas()$ utilizamos esta matriz de "*voxeles*" para crear un cubo en cada punto apropiado a su altura correspondiente dado el índice que tenía la matriz de voxeles.

El gradiente de temperaturas se creó con la función $fngradiente()$ la cual crea una flecha por cada punto que no está en los muros. La función que crea las flechas $createFlecha(i, j, ang, z, h, r)$ recibe como parámetros la posición de donde sale la flecha (i,j), el ángulo de inclinación, el cual lo obtenemos de una matriz que calcula y guarda estos ángulos, dados los valores de los gradientes en x e y, ya calculados en el primer programa, recibe además la altura a la cual queremos ver dibujadas las flechas, el valor que se utilizó para la discretización y el largo de la flecha.

Cada una de estas visualizaciones (el gradiente y las curvas de nivel) fueron creadas en una sola gpu, juntando cada cubo/flecha con la función merge.

En este último programa, se puede caminar por el hotel, utilizando las flechas arriba, abajo, izquierda y derecha. Se visualiza el gradiente de temperaturas al presionar la tecla X (tuve un problema con control-derecho que se pedía en el enunciado; se ejecutaba la acción de mostrar el gradiente sin la necesitad de apretar la tecla derecha, ie, solo apretando control y además si la apretaba ambas teclas la cámara se giraba, entonces cambié a la tecla X) y las curvas de nivel al presionar espacio.

Instrucciones de ejecución

Las instrucciones de ejecución del programa $hotel - solver.py$ son: $pythonhotel - solver.pyhotel.json$. hotel.json es un archivo el cual tiene la información de las medidas del hotel, ya sea el ancho y largo de su pasillo, de las piezas y de los muros, además el nombre $filename$ con el cual guardaremos la matriz de temperaturas, el valor de $window_loss$, $ambient_temperature$ y de $heater_power$, además de el vector en $windows$ el cual hace referencia con un 0 a cuando una ventana está abierta y con un 1 a cuando está cerrada.

Y el programa $hotel - viewer.py$ se ejecuta de la misma manera que el anterior solo cambia el nombre del programa.

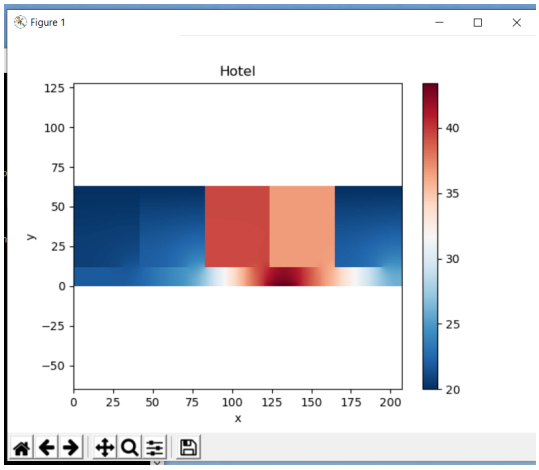
Retornos

El programa $hotel - solver.py$ presenta de manera gráfica la temperatura al interior del hotel, mientras que $hotel - viewer.py$ presenta un dibujo del hotel en 3D, mostrando en el suelo las temperaturas, yendo desde el color blanco-amarillo-naranjo-rojo en orden ascendente de temperaturas, además da la opción de visualizar el gradiente y las curvas de nivel, finalmente, retorna los colores utilizados en el suelo y las temperaturas que tienen asociadas.

Screenshots:

Capturas de las intrucciones de ejecución y de lo que entrega el programa: *hotel − solver.py*

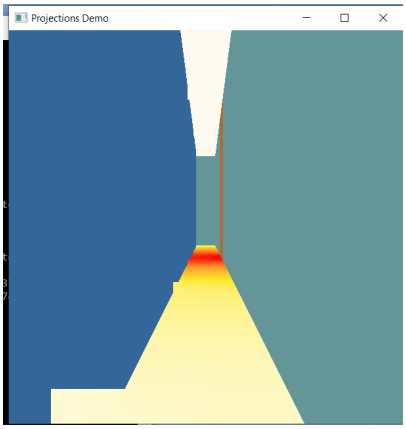
```
cg) C:\Users\julia\Desktop\paredes-quiroz_julia-javiera\Tarea3b>python hotel-solver.py hotel.json
```



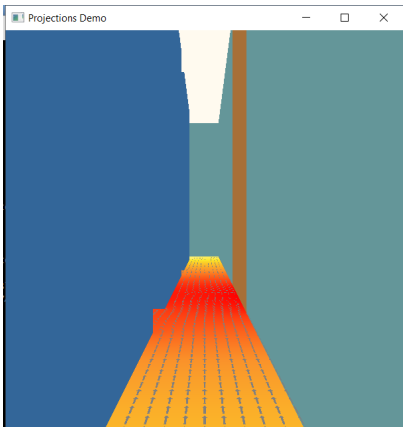
Captura de la instrucción de ejecución del programa *hotel − viewer.py* y el retorno de los colores y sus temperaturas:

```
(python-cg) C:\Users\julia\Desktop\paredes-quiroz_julia-javiera\Tarea3b>python hotel-viewer.py hotel.json
blanco:  [1, 1, 1]  temperatura:  20.0
amarillo:  [0.9882352941176471, 0.9098039215686274, 0.1568627450980392]  temperatura:  27.81518253721067
naranjo:  [0.9803921568627451, 0.5725490196078431, 0.16470588235294117]  temperatura:  35.63036507442134
rojo:  [1, 0, 0]  temperatura:  43.44554761163201
```

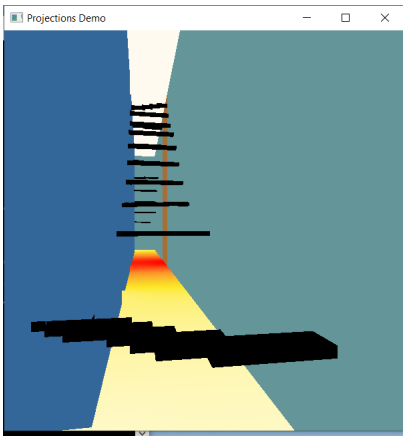
El hotel a primera vista:



Visualización del gradiente de temperaturas:



Visualización de curvas de nivel:



Visualización del gradiente y las curvas de nivel: