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

Reporte de documentación

Tarea 1C: Pixel-paint

Alumna: Julia Paredes

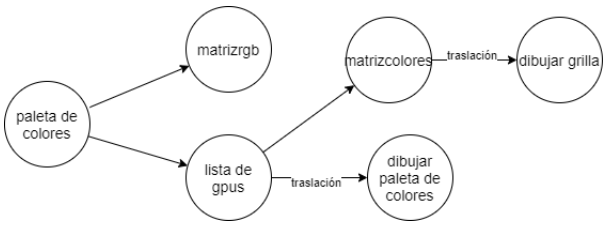
RUT: 20.240.283-6

Profesor: Daniel Calderón

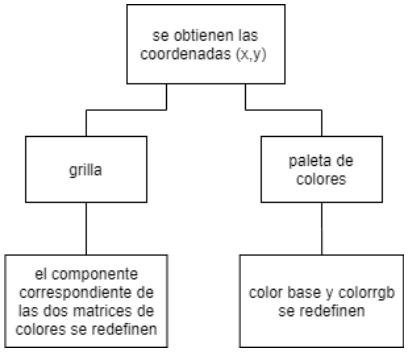
Auxiliares: Alonso Utreras - Nelson Marambio

Fecha de entrega: 9 de Mayo, 2020

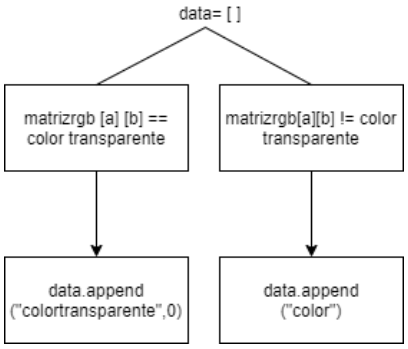
Lo primero a programar fue la grilla...



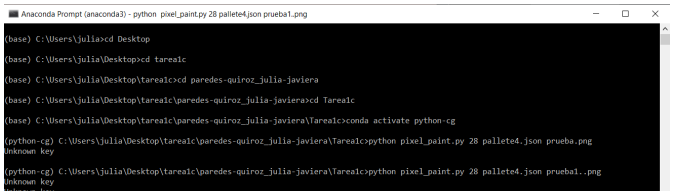
Explicando el diagrama:
Primero se crea una lista compuesta de GPUs, específicamente hay un cuadrado (*createQuad*) de cada color presente en la paleta dada. Esto se realiza con la función *createallcolors*. El elemento a_i de esta lista será dibujado a la izquierda del programa, cada traslación será producida por el elemento b_i de lista retornada de la $funcionrango$. De esta manera se muestra la paleta de colores.
Luego se crean dos matrices de tamaño NxN, una compuesta de GPUs y otra que contiene los colores de la forma [r,g,b], en un comienzo ambas están solo compuestas de cuadrados del color transparente/color transparente . La primera de ellas se llama *matrizcolores*, más adelante se dibuja cada componente de esta matriz, y para darles una posición correcta, se les aplica una traslación en base a la lista retornada de $funcionrango$. La segunda matriz se le llama *matrizrgb*, la cual en un futuro se utilizará para guardar la imagen creada.
La función $funcionrango$ se encarga de retornar una lista con las posiciones (de un eje) que debería tener cada pixel de la grilla. Se utiliza la función np.arange, por ende en aquella función solo se define el inicio y distancia entre el centro de los pixeles (estos valores son calculados en función de N). Se acota (predeterminadamente) como final 0.6, pues en el programa los últimos 0.4 serán utilizados para colocar la paleta de colores, y cuando se utiliza esta función para dibujar la paleta, se redefine final = 1.



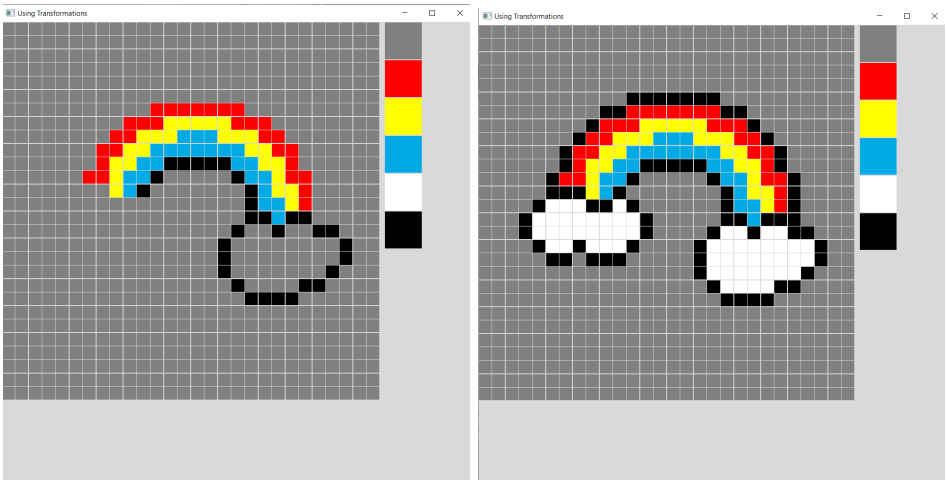
Este segundo diagrama muestra lo que sucede cuando se hace click, en primer lugar se guardan en las variables a y b las posiciones en x e y en una escala [0,N] considerando únicamente los primeros 800 pixeles del programa, que es donde esta ubicada la grilla. De esta manera; si hago un click en la grilla se modifica el color del cuadrado ubicado en matrizcolores[a][b] y también se modifica el elemento a,b de la matrizrgb. El caso de clickear la paleta de colores, se crea una variable c que convierte el eje y de la posición del click en una escala [0,10] dado que hay dos columnas como máximo con colores de la paleta, de 10 cada una. Así puedo tomar ese color desde la lista de GPUs creada en un comienzo y desde la lista de colores ([r,g,b]) dada en la paleta y tener el color seleccionado como color a utilizar en la grilla.



Por último este diagrama representa la forma de guardar el dibujo creado; Primero se crea una lista vacía llamada data, a la cual se le van agregando los elementos de la matrizrgb, y si uno de estos es el color llamado transparente, al momento de añadirlo a la lista, se le agrega el 0 como cuarta componente. Después se crea el data de la matrizrgb, la imagen jpeg de este data, con la intención de tener una imagen la cual transformar, por ende luego se transforma a RGBAz se le cambia el data por el creado en un principio. Con esto se produce una reflexión y rotación, las cuales se arreglan con ...mirror y ...rotate. Finalmente se guarda la imagen con el nombre del tercer argumento, en formato png.
Hablando de los argumentos; el primero de ellos es un número (al cual llamamos N), que corresponde al alto y ancho de la grilla cuadrada en la cual se dibujará. El segundo es la paleta de colores a utilizar(ya creada), debe estar en formato json (en la carpeta envío 4 que fueron mis pruebas). El último argumento es el nombre con el cual se guardará la imagen creada. Este nombre debe mencionarse con fomarto png, es decir *nombredelarchivo.png*.
En el programa, solo funciona el click izquierdo, el cual puede ser usado para seleccionar el color y pintar, ya sea cuadro por cuadro o varios juntos arrastrando el mouse. Para pintar, basta hacer click sobre el color elegido y luego dibujar en la grilla, hay que considerar que si no se hace click sobre el color elegido, el color que viene por defecto es el mismo color que tiene la grilla, que corresponde al color que quedará como transparente. Por esto mismo si al dibujar, se equivocan y seleccionan un pixel que no corresponde, pueden volver a pintarlo con otro color o volverlo al color transparente, simplemente seleccionando el color correspondiente y volviendo a pintar aquel pixel.
Mientras se ejecuta el programa, las teclas S y G sirven para guardar la imagen que se dibujó. Y si después se realizan más cambios, se puede volver a presionar algunas de ellas y la imagen se actualizará.
Para finalizar mostraré screenshots del programa.



La última linea fue lo que escribí para producir lo siguiente:



Y al momento de apretar la tecla S o G, se guardó en la carpeta donde está ubicado el programa, con el nombre del tercer argumento: "prueba1.". Como se puede observar, el programa funciona para N $>$ 16.