1 Introduction

In this homework assignment, you will use C++ to create and manage geometry which you will then pass to Vulkan for drawing lines. Managing geometry is a fundamental task in any graphics application. For this assignment, you will generate all required geometry using simple transformation rules. In particular, you will draw a space-filling curve known as the *Hilbert curve*. The construction rules for this fractal curve are explained in more detail later.

Vulkan provides various ways to draw lines. For this assignment, you do not need to worry about the Vulkan setup needed to draw lines. Starter code is provided that will draw lines for you. You will however need to organize your line geometry data. The starter code provides some guidance on how to organize your data and send it to Vulkan for drawing.

1.1 Hilbert Curve

Discrete approximations to the 2D Hilbert curve are used to map data from a 2D space (e.g. an image) to a 1D index space. The good thing about this mapping is that it preserves locality - points that are close in 2D space are also close in index space thus leading to better cache locality when storing 2D data in memory. Fig. 1 shows an example of a *discrete* Hilbert curve on a 2D grid.

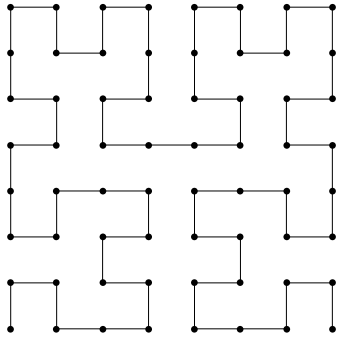


Figure 1: A discrete Hilbert curve on an 8×8 square grid.

1.2 Construction

There are different ways to generate a discrete Hilbert curve; for this assignment we will focus on a recursive geometric approach. Given a $2^n \times 2^n$ grid ($n \geq 1$) within the unit square $[0,1]^2$, a Hilbert curve that visits all 2^{2n} points on the grid can be obtained by starting with a Hilbert curve defined on a $2^{n-1} \times 2^{n-1}$ grid and refining it by applying appropriate affine transformations. Fig. 2 illustrates this idea. The base case ($n = 1$) consists of the corners of the unit square that are visited in the order: $(0,0) \rightarrow (0,1) \rightarrow (1,1) \rightarrow (1,0)$. In order to obtain the curve at the next level ($n = 2$), we scale down the the curve from the previous level ($n = 1$) and place four copies of it on the grid; two copies go to the top-left and top right while two *rotated* copies go to the bottom-left and bottom-right. We can then simply connect these copies together as shown in Fig. 2. This process can be repeated to obtain subsequent levels. In the limit (as $n \rightarrow \infty$), the curve obtained is the continuous Hilbert curve that fills the unit square.

The exact transformations that you need are not spelled out here but can be inferred from Fig. 2.

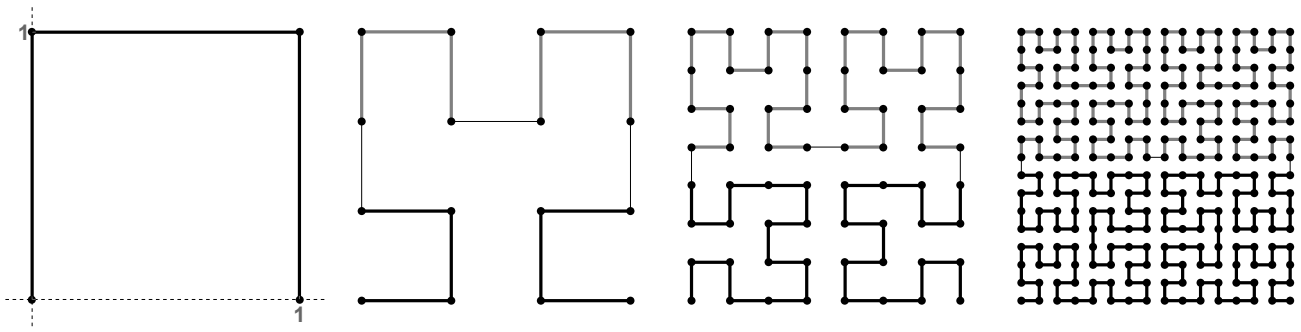
2 Requirements

Your task is to write a C++ program that procedurally generates the geometry associated with the Hilbert curve. Your program should then use Vulkan to draw the geometry on the screen. The number of iterations should be a user adjustable parameters. Your program should be able to — through a keyboard control — produce and display the Hilbert curve at progressively increasing levels of detail.

2.1 Functional Requirements

Your program should have the following functionality:

- Generate and display the line segments associated with the discrete Hilbert curve for $n = 3$. (8 points)



(a) $n = 1$ (b) $n = 2$ (c) $n = 3$ (d) $n = 4$

Figure 2: The first four iterations of the Hilbert curve. Each iteration consists of four scaled-down versions of the previous iteration.

- Provide keyboard controls that allow the user to display the curve at different levels of levels of detail, starting from the level ($n = 3$), down to the the base level $n = 1$ and up to the maximum level that the window can support. The user should be able to increase/decrease the level one step at a time.

  To keep things simple, for this assignment, we'll be using a fixed window of size 1024×1024. (12 points)

2.2 Non-Functional Requirements

Please also keep the following non-functional requirements in mind:

- It is highly recommended that you build your program on top of the provided starter code. The TAs will be going over the starter code and providing additional helpful tips during tutorials. Please start early and seek help in a timely manner in case anything is not clear.
- You can use the mathematics library GLM to perform transformations.
- You can use the Standard Template Library (STL) to manage geometry.
- Your program should compile and run on the machines in MS 239.

2.3 Demo (5 points + 3 bonus)

You will be required to demo your program to your TA during a scheduled tutorial session in MS 239. Please be prepared to walk your TA through your implementation during the demo.

You can earn bonus points by demoing early.

3 Submission and Grading

This is an individual homework assignment. Please make sure that you are aware of what *is* and what *is not* allowed. Refer to the academic misconduct section on the course website for details. If in doubt, please consult the instructor for clarification.

Please submit via Gradescope. Please take note of the following when preparing your submission.

- Please submit only the files you have modified from the provided starter code as well as any other helper files needed to compile and run your program.
- You do not need to submit any modifications to VulkanLaunchpad.
- Please include a README file with instructions on how to compile, run and test your program. The build instructions can be as simple as stating that you are following the same build procedure as the starter code. However, if your program requires a different build procedure, please provide clear and detailed instructions on how to compile and run your program.
- Additionally, use the README file to cite any resources that you used.

4 Further Reading

- Hilbert curve on Wikipedia:
  `https://en.wikipedia.org/wiki/Hilbert_curve`.
- Hilbert curve on Wolfram MathWorld:
  `http://mathworld.wolfram.com/HilbertCurve.html`.