OVERVIEW:

- This an iOS App developed using Swift and GLSL in Xcode.
- It draws Bézier Spline with every touch on the screen. Based on the concept of Bézier Spline, as soon as there are 3n+1 control points on the screen it draws n splines. These control points can be selected and moved to any other position on screen. Other parts of the spline will change according to the translation without breaking the spline.

EXECUTION :

- Open the .xcodeproj file in Xcode.
- Run the project after setting the active scheme as iPhone6
- The can be directed to an actual handset by selecting the connected iPhone as the active scheme.

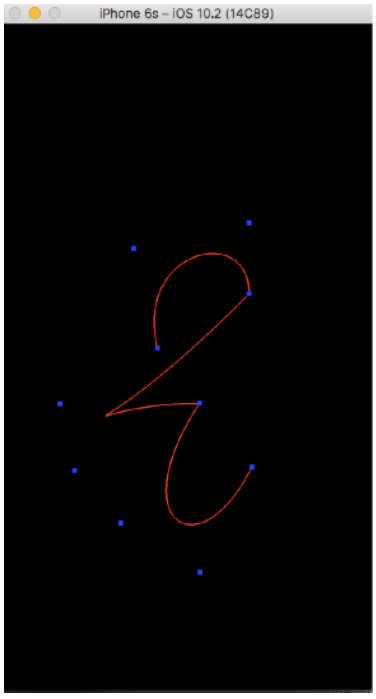
BEHAVIOR:

- When the app launches in simulator, any tap on the screen will add a control point (using GL_POINTS) at the tapped coordinates on screen.
- As soon as there are 4 (3*1+1) control points, first Bézier Spline will be drawn using the linear interpolation equation. Second spline segment will be drawn when there are 7(3*2+1) control points and so on.
- All four points are interpolated and the new calculated points (about 100 points for t varying from 0 to 1) are passed to vertex shader with primitive type as GL_LINE_STRIP.
- This Spline is flexible and any control point can be grabbed and moved/translated. The new set of points are again interpolated and the spline is redrawn without being broken..
- All the calculations including interpolation are done at CPU side.

THE CODE

- In Xcode go to the GameViewController.swift from the Project Navigator. This file has the 90% of the logic including touch-listeners and interpolation function.
- After all the touch inputs and interpolations, this file passes the uniform variables and position attribute values to vertex shader.
- The vertex shader (written in GLSL) calculates the orthogonal projection matrix and multiplies it to the vertex coordinate passed by the CPU code and sends them to the fragment shader.
- Fragment shader has been kept simple for this project and does not include any extra functionality.

SCREENSHOT:



DEPENDENCIES:

- The dependencies are: it might not draw the solid lines for some models other than iPhone6.

LIMITATIONS:

- In this project no matrix transformations have been used. To implement translation of control points, traditional insert and delete functions have been used on vertex arrays. Hence greater part of the work is being done by CPU which can be delegated to GPU using matrix transformations.
- Interpolation is done on CPU side which can also be delegated to GPU.

: