# Simulating a Flight Path Over Terrain

Pramod Rao, Kaustubh Sakhalkar, Linjie Lyu

# Overview

→Introduction←

Terrain Generation

Flight path

Camera Views

# Introduction

http://antonior-software.blogspot.com/2017/03/simple-animation-interpolation.html

https://www.studioghibli.com.au/

https://jsfiddle.net/franciscop/jsfv13no/

# Our Project

- Generate a Terrain
- Find shortest path for given points on terrain
- Interpolate a smooth curve for the shortest path
- Top view (Landscape) and observer view of flight

# Overview

Introduction

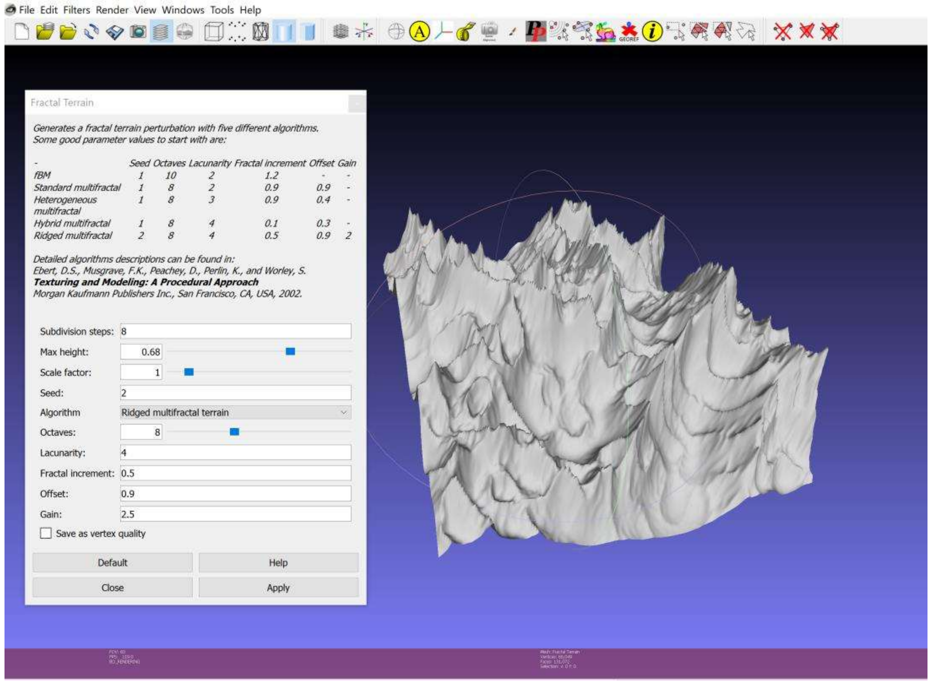
→Terrain Generation ←

Flight path

Camera Views

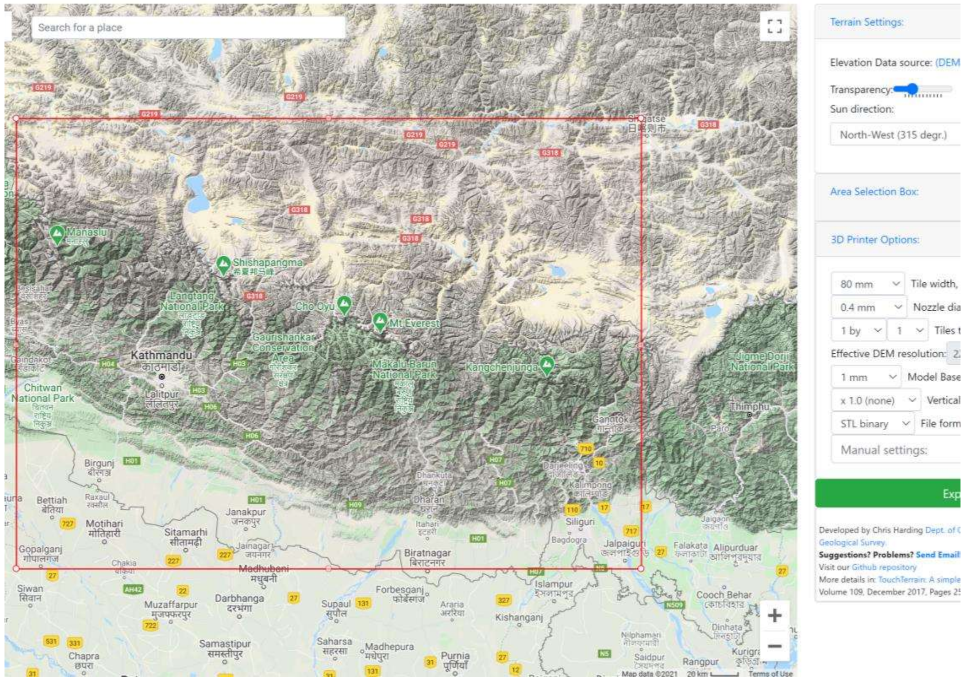
# Terrain Generation

## Fractal Terrain




Meshlab

## Real World Terrain




https://touchterrain.geol.iastate.edu/, Iowa State University

# Overview

Introduction

Terrain Generation

→Flight path←

Camera Views

# Travelling Salesman Problem

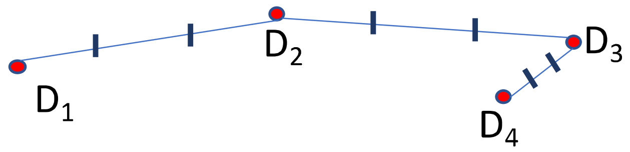
- Use a genetic algorithm for finding the shortest path.
- Genetic algorithms are inspired by evolution ("survival of the fittest").
- We give a bird's eye view of the algorithm.

<u>Algorithm</u>

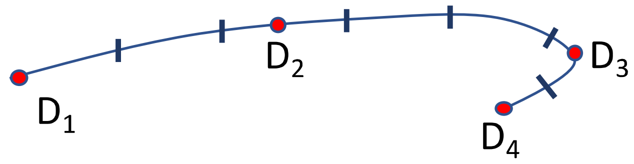
1. Initialize paths randomly
2. Determine distance matrix
3. Repeat until stopping criterion:
    1. Select parents.
    2. Perform crossover and mutation.
    3. Calculate the fitness of new population.
    4. Append it to gene pool.

Code Credits: [link]

# Chord Length Method




Sharp kinks at interpolation points if curve follows too closely to the polygon




Choose number of sample points according to chord length ratios

$$L = \sum_{i=1}^{n} |D_i - D_{i-1}|$$

$$L_k = \frac{\sum_{i=1}^{k} |D_i - D_{i-1}|}{L}$$

$$\implies t_0 = 0$$

$$\implies t_k = \frac{1}{L}\left(\sum_{i=1}^{k} |D_i - D_{i-1}|\right)$$

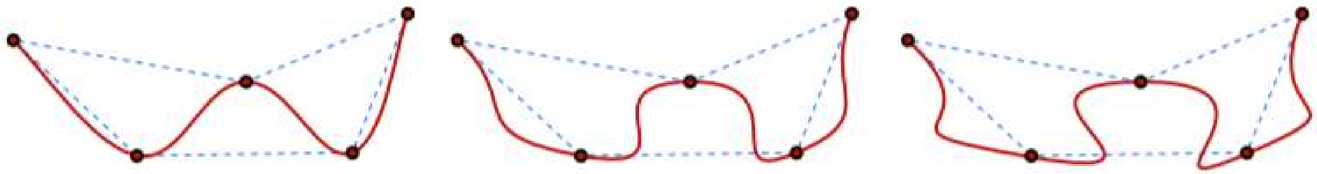
$$\implies t_n = 1$$

https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/INT-APP/PARA-chord-length.html

# Catmull-Rom Splines

- Tangent at each point $D_i$ is calculated based on $D_{i-1}$ and $D_{i+1}$
- Adjustable tension parameter $\tau \in (0,1)$ which controls the "tangentness" of the curve.

$$\mathbf{p}(s) = \begin{bmatrix} 1 & \mathbf{t} & \mathbf{t}^2 & \mathbf{t}^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau - 3 & 3 - 2\tau & -\tau \\ -\tau & 2 - \tau & \tau - 2 & \tau \end{bmatrix} \begin{bmatrix} \mathbf{D}_{i-2} \\ \mathbf{D}_{i-1} \\ \mathbf{D}_{i} \\ \mathbf{D}_{i+1} \end{bmatrix}$$



$\tau = 0.2$ $\tau = 0.5$ $\tau = 0.75$

Credits: https://www.cs.cmu.edu/~fp/courses/graphics/asst5/catmullRom.pdf

# Overview

Introduction

Terrain Generation

Flight path

→Camera Views←

# Curves – B-Splines

- Chordal parameterization

- Not-a-knot end condition:

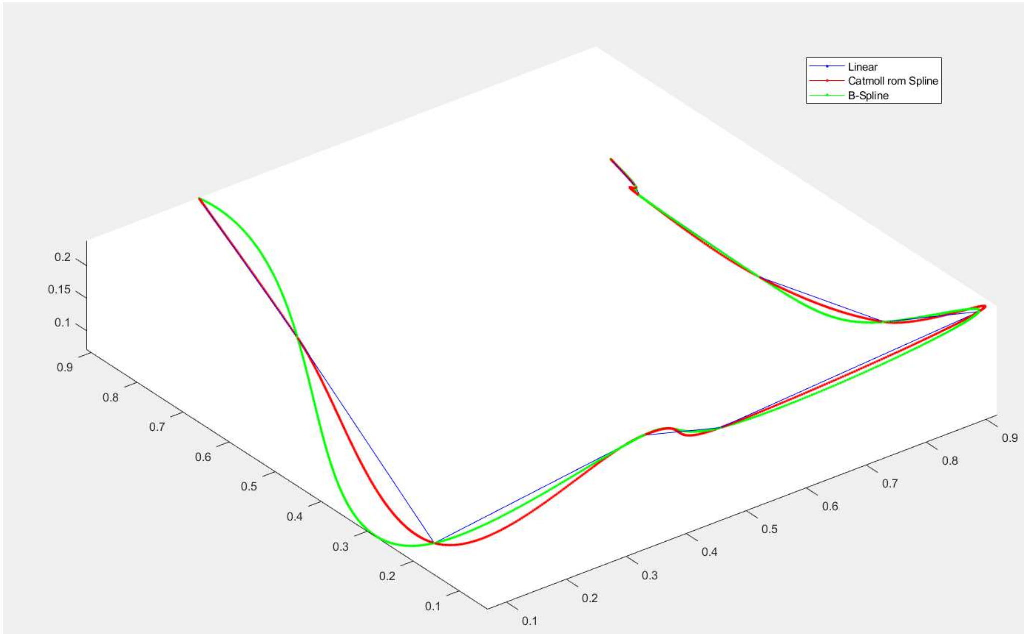
  The third derivative

  is also continuous at the $2^{nd}$

  and the second last points.

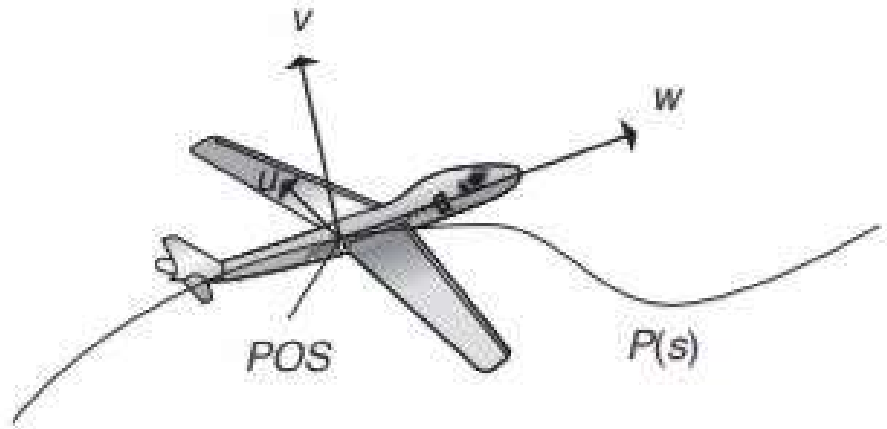
- Continuity: $C^2$

  Not that short.

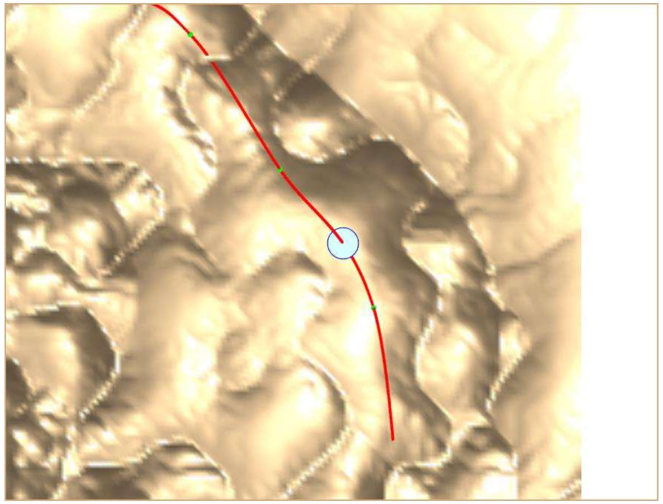



Blue : Linear
Red : Catmull Rom
Green : B-Spline

# Camera Views




- **Observer View:**
  - Position: S(t)
  - Up: [0,0,1]
  - Front: S'(t)
  - Left: Up × Front



- **Landscape View:**
  - Position: Above the plane with a constant height.
  - Fixed camera pose.

# Fly-Over-Terrain

- Pre-load the terrain and plot the path.
- Iterate samples on the curve representing the movement.
- Update the camera position and direction in every frame.
- Speed control.

```
for i = [1:size(curve, 1)-5]
    set(terrain_marker, 'Xdata', curve(i,1));
    set(terrain_marker, 'Ydata', curve(i,2));
    set(terrain_marker, 'Zdata', curve(i,3));

    % Observer view: perspective
    campos(obs_view, curve(i,:));
    camtarget(obs_view, curve(i+5,:));
    % Landscape view: orthographic
    campos(land_view, [curve(i,1:2), camLandscapeHT]);
    camtarget(land_view, [curve(i,1:2), 0]);

    camlight(head_light,'headlight');
    drawnow;

    distance = norm(curve(i+1)-curve(i,:));
    const_speed = distance/anime_speed;

    pause(const_speed);
end
```

# Demo

"Talk is cheap. Show me the code."

-Linus Torvalds

# Questions?

Thank you for your attention.