Perlin Noise II

CMSC425.01 Spring 2019

Administrivia

- Final project
  - Update for Monday (need to verify group membership for Elms)
  - Final project deliverables (rubric to come)
  - Final project demonstration schedule (May 20, 1:30-3:30)
- Final homework (Hw3)
  - Out now, due Sunday May 5th
  - Solutions and review Monday May 6th
- Final midterm
  - Thursday May 8th

Unity VR job

- ONE YEAR Temporary role- Great way to gain experience in VR!
- Space Telescope Science Institute is seeking a temporary Unity3D software developer to develop simulations and scenarios in virtual reality (VR). The successful applicant would work with experts in astrophysics and public outreach to develop immersive astrophysical scenes. No prior knowledge of astrophysics is required. We utilize accurate 3D models of space telescopes and simulated environments to interact with in real time. The Unity3D developer will be building the simulation apps for the HTC Vive and Oculus platforms, but may explore other platforms as required.

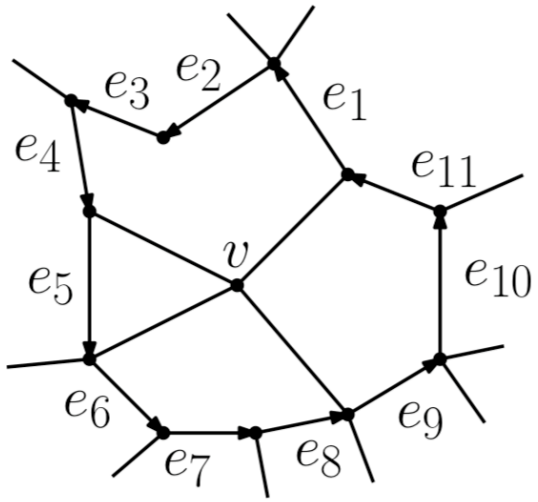
- As a part of the VR team, you will be:
  - Pioneering the future of astronomy learning and outreach by working on cutting-edge technologies
  - Building Virtual simulations and experimental prototypes
  - Working in C# and Unity3D to create clean, organized, optimized, reusable code modules that can be adapted for multiple projects
  - Working with multiple 3rd Party SDKs to integrate VR tracking, computer vision, and haptics into Unity
  - Building amazing 3D simulations using the latest in VR/ AR/MR technology

- Basic Qualifications:
  - Bachelor’s degree in Computer Science, Human Computer Interaction, or related field experience
  - Experience developing for Virtual Reality technologies.
  - Experience working in Unity3D and C#
  - Experience implementing 3D games and/or 3D simulations in Unity3D
  - Experience working with animated and rigged 3D models in Unity3D
  - Experience with 3D user interface design
  - Strong knowledge of OOP, design patterns, event delegates, asynchronous functions, data structures and algorithms
  - Experience working with a team in a professional capacity,
  - High standards for work quality, self-motivated, able to work with minimal supervision

- Desired Skills:
  - Experience with rendering pipelines on different platforms, profiling and optimization techniques in VR
  - MS degree in Computer Science, Human Computer Interaction, or related field experience
  - Experience with 3D Graphics
  - Experience with Shader Programming
  - Experience with 3D modeling tools (ex: Maya, 3D Max, or Blender)
  - Experience with Unity3D interfacing other external data elements
  - Experience with Augmented Reality and Android / iOS development a bonus

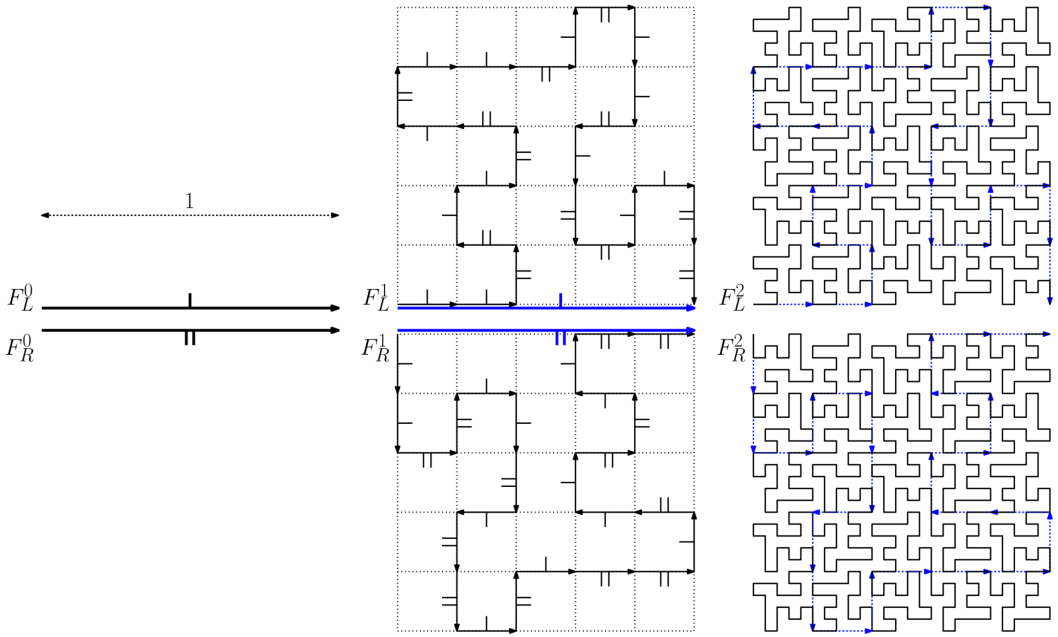
Why care about this problem?

Given vertex v in a cell complex of a 2-manifold, the *link* of v is defined to be the edges that bound the faces that are incident to v, excluding the edges that are incident to v itself. Present a procedure (in pseudocode) that, given a vertex v of a DCEL, returns a list L consisting of the half edges of v's link ordered counterclockwise about v. For example, in the figure below, a possible output would be $\langle e_1, \ldots, e_{11} \rangle$. (Any cyclic permutation would be correct.)



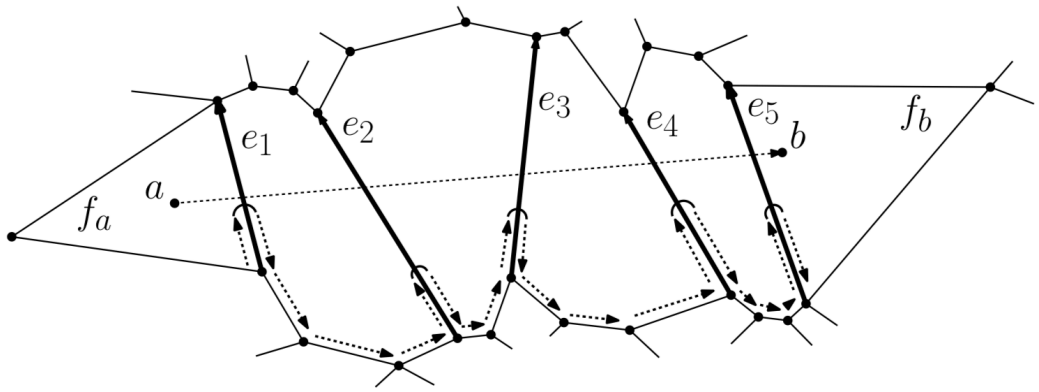
Problem – Fractal curve

- Derive an L-system that generates FL and FR. In particular, please provide the recursive rules for FL and FR.
- Consider the curve FL in the limit. Derive its fractal dimension.
- Each generation distances are scaled by σ = 1/5, and each individual segment of the basic length is replaced by 25 segments of the next smaller size.



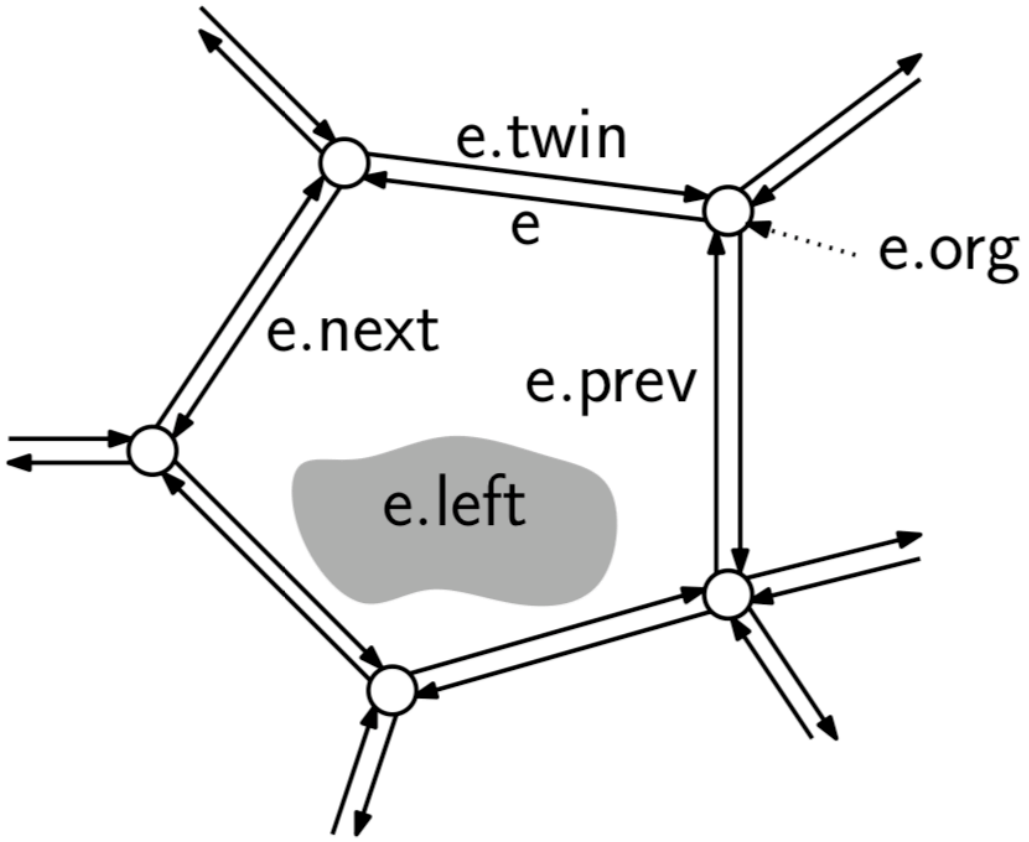
Problem – DECL intersection

- Compute a list L = ⟨e1,e2,...,em⟩ of edges that intersect a line segment **ab**
- Given:
  - Faces fa and fb that contain a and b, respectively
  - Function e.cross(a,b) that returns true/false if edge e crosses **ab**



Winged edge representations

- Vertex v has coordinates plus one link to incident edge
- Face f has link to one half edge
- Edge (origin u, destination v) has
- *e.org*: e's origin
- *e.twin*: e's opposite twin half-edge
- *e.left*: the face on e's left side
- *e.next*: the next half-edge after e in counterclockwise order about e's left face
- *e.prev*: the previous half-edge to e in counterclockwise order about e's left face (that is, the next edge in clockwise order).



Today’s question

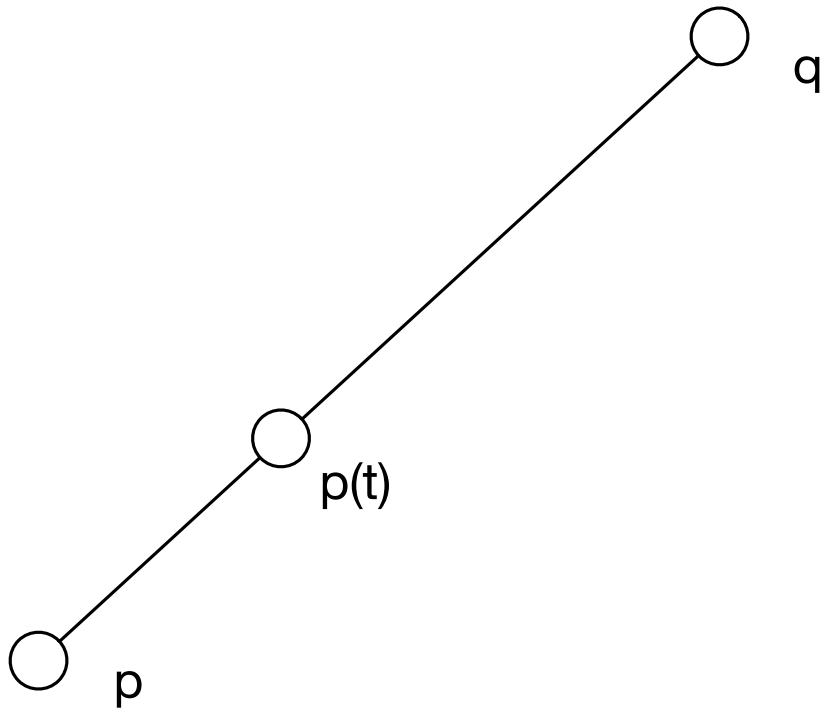
Perlin noise in 2D?

Parametric line segments

$$p(t) = p + tv$$

with $v = q - p$

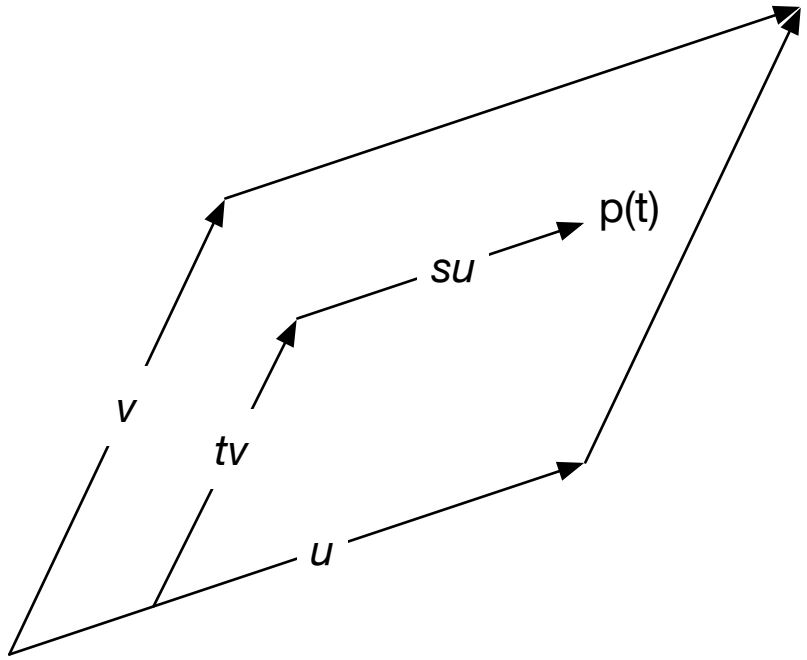
```
for t = 0 to 1 by deltat
     x = px + t * vx
     y = py + t * vy
     plot(x,y) // or line(lx,ly,x,y)
```



Parametric planar patches

$$p(s,t) = p + tv + su$$

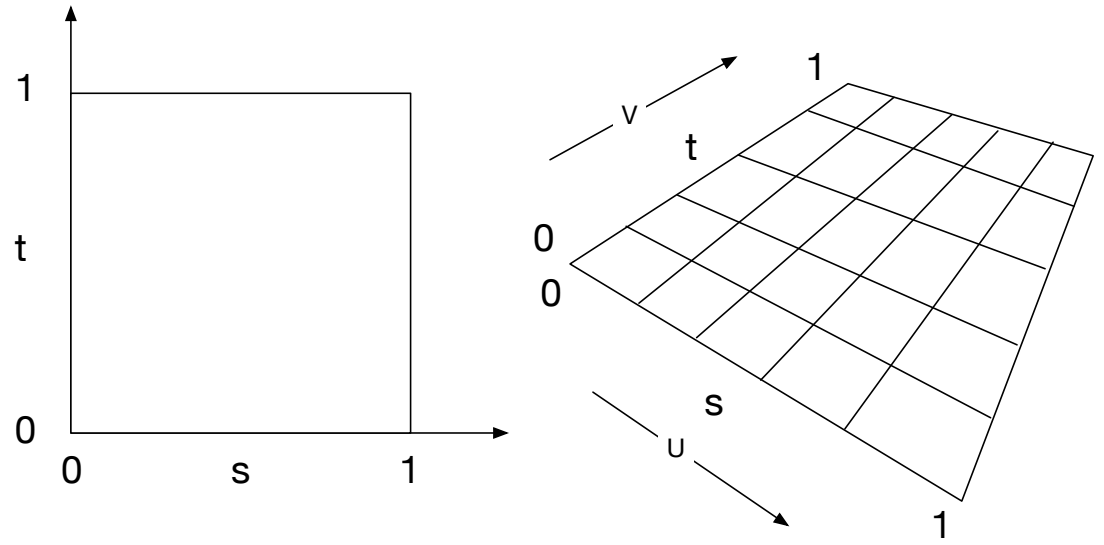
```
for t = 0 to 1 by deltat
     x = px + t * vx + s * ux
     y = py + t * vy + s * uy
     z = pz + t * vz + s * uz
     plot(x,y,z)
```



Creating planar mesh

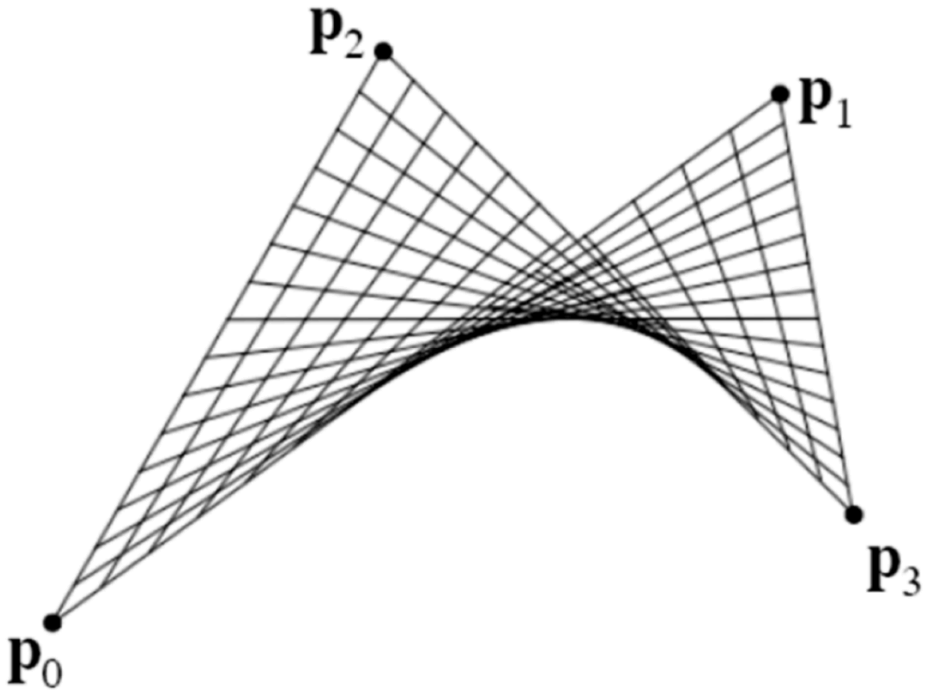
- How create mesh data structure from parametric patch?

- List of vertices
- List of edges
- List of faces



Bilinear patches and interpolation

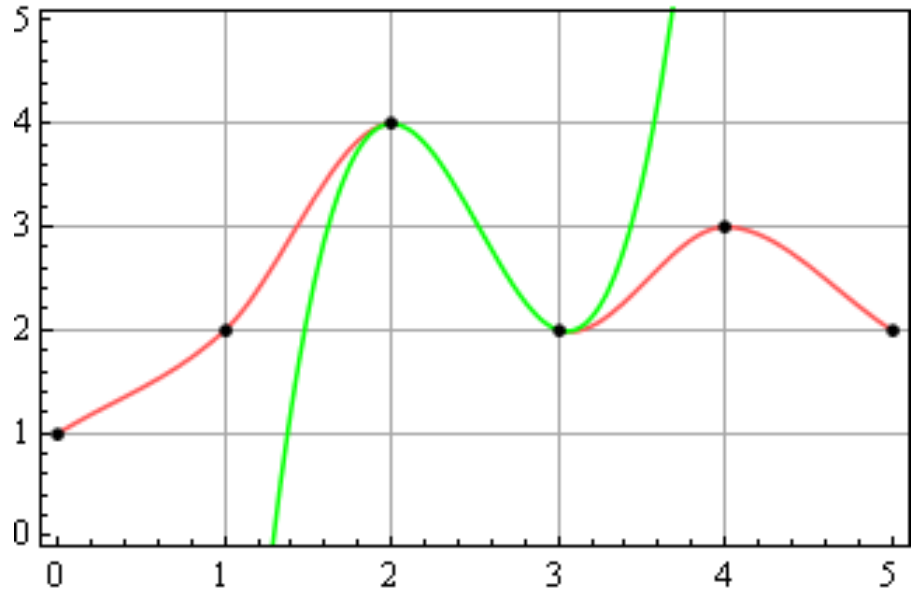
- Interpolation of four points
  - May not be co-planar
- Ruled surface – swept out by straight line
- Will develop equations in class



Cubic interpolation

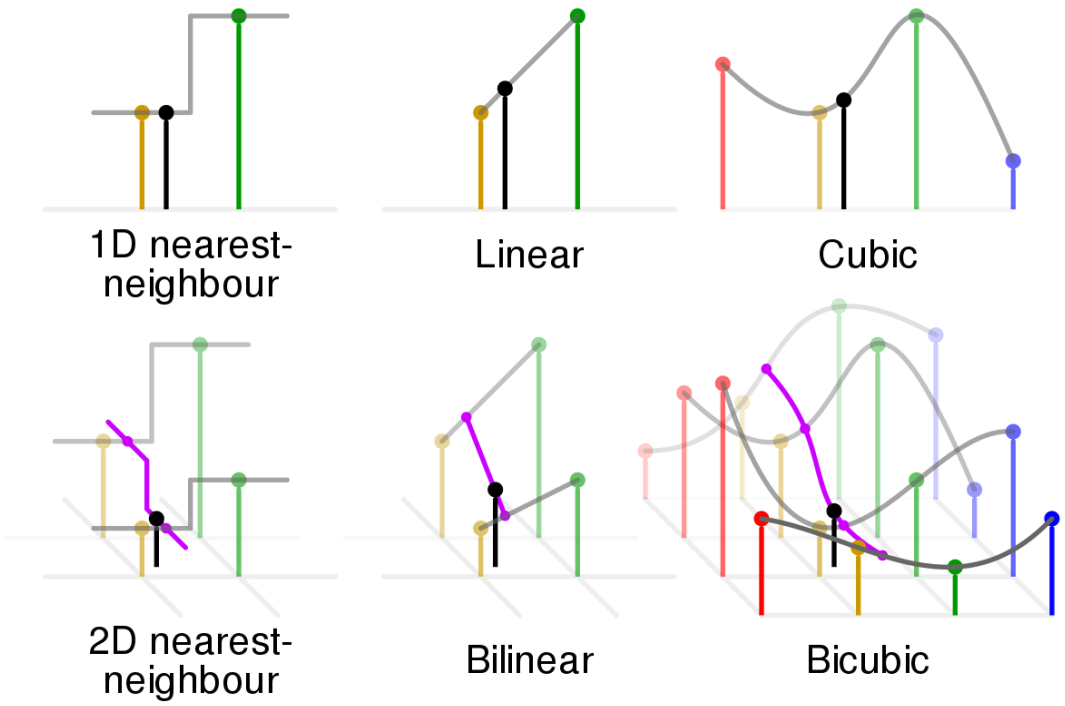
- $P(t) = ax^3 + bx^2 + cx + d$

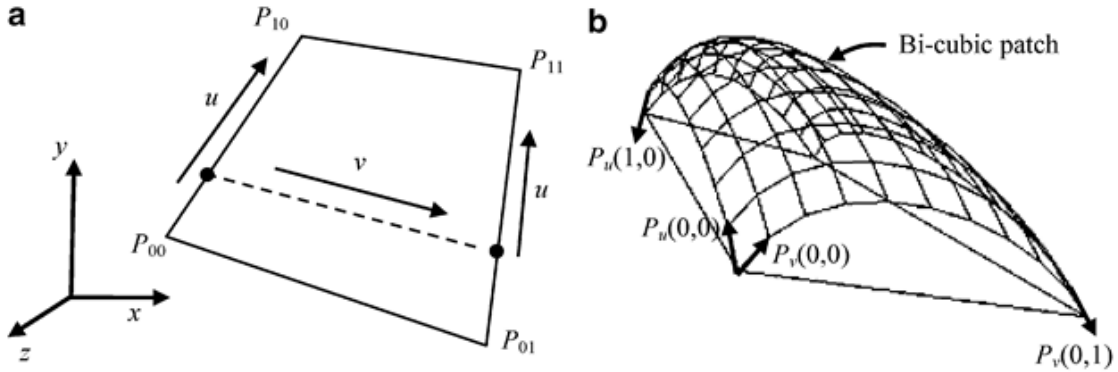
- Can match tangents at ends
- Good enough for human eye

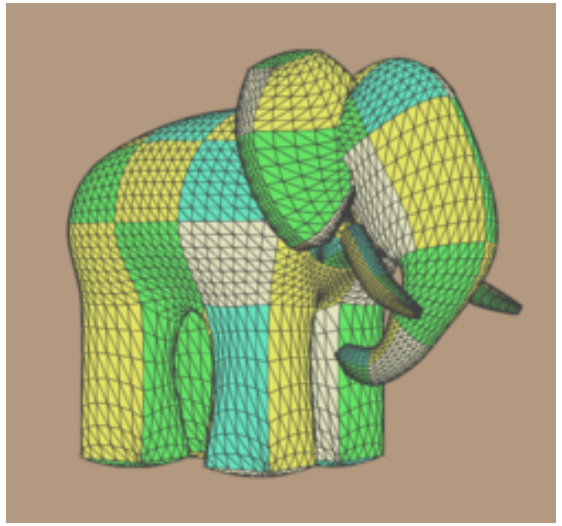


Bicubic surface patch

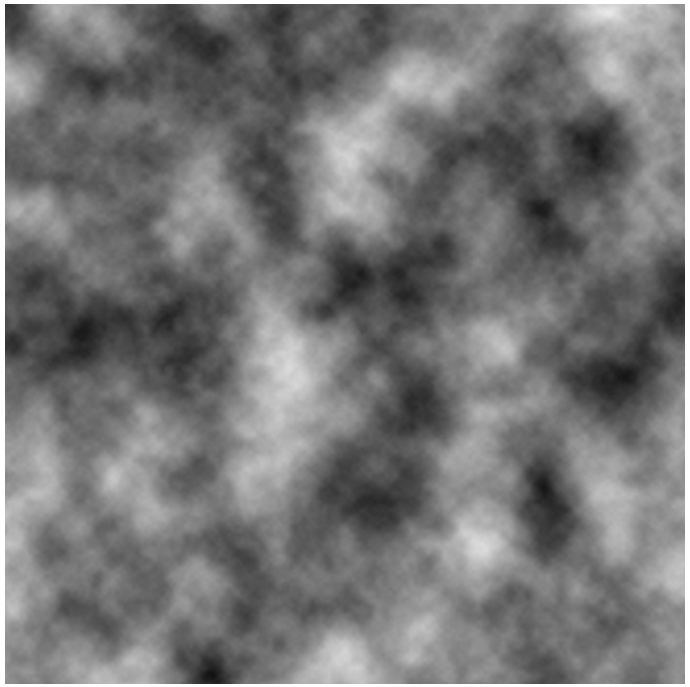
- Cubic curve in both directions



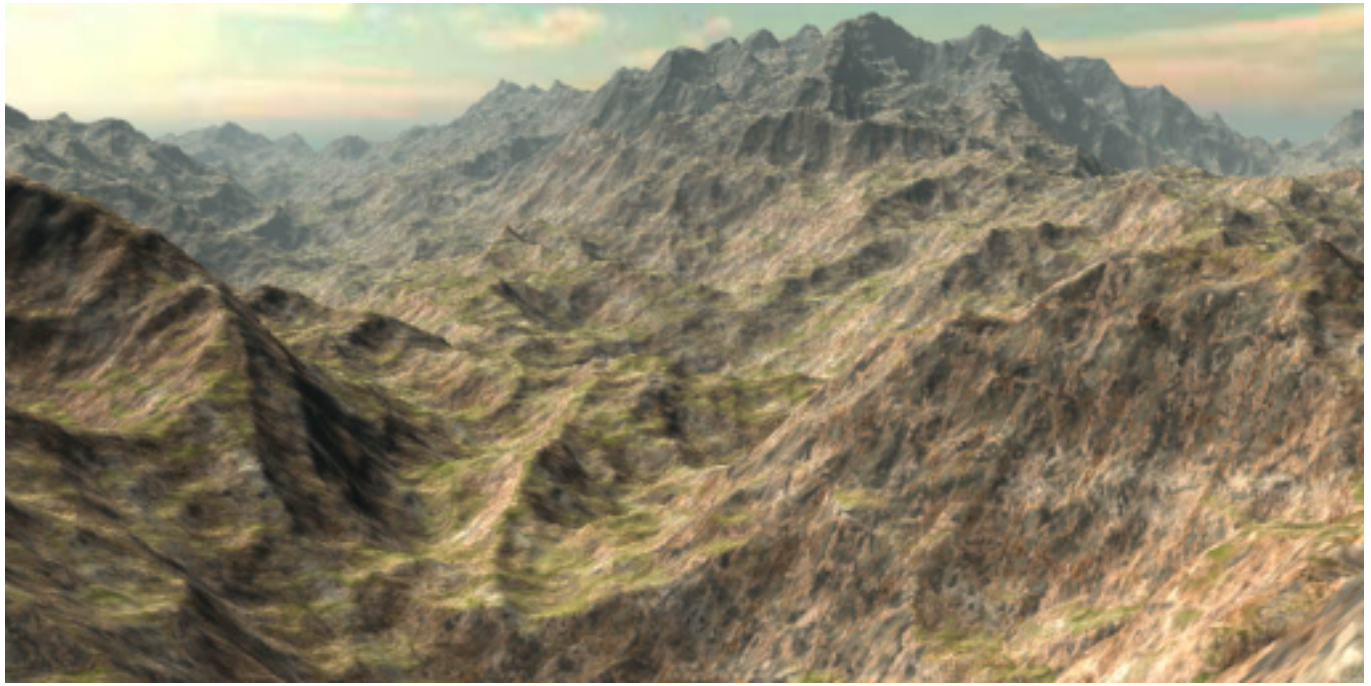




2D Perlin noise



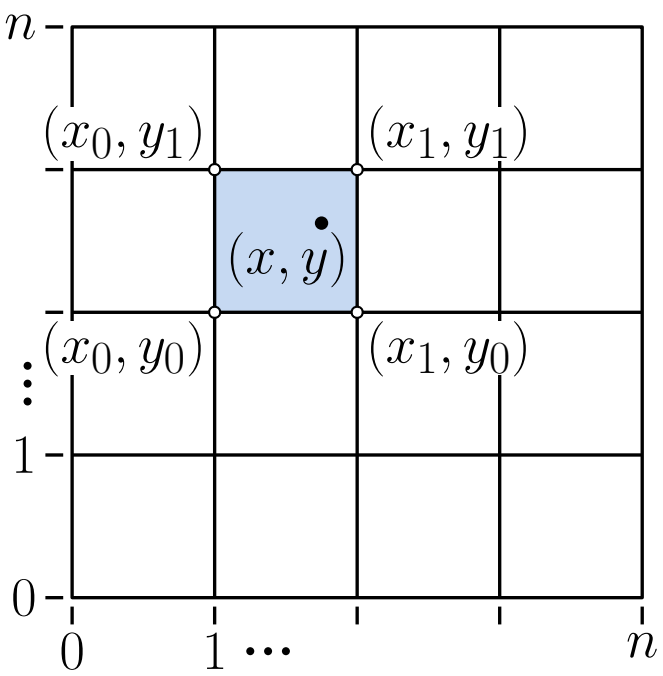
(a)



(b)

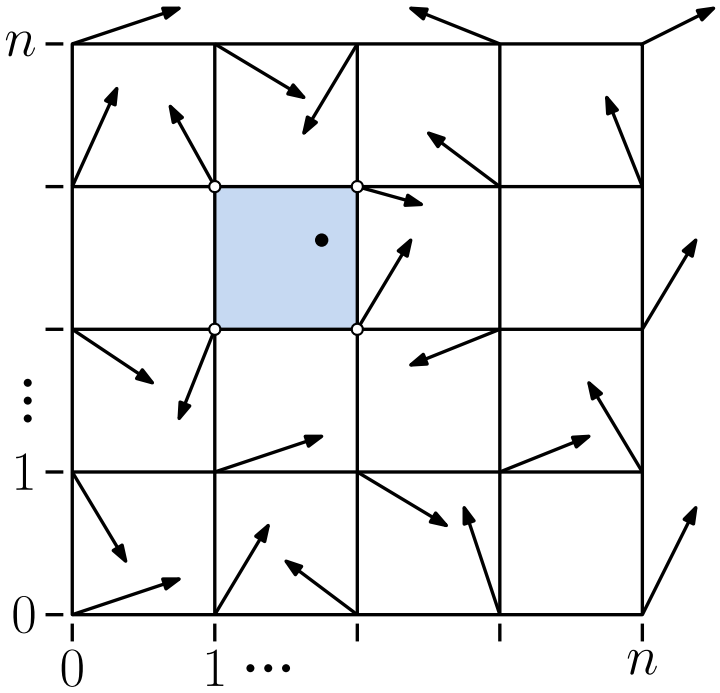
2D Perlin algorithm

Initial grid



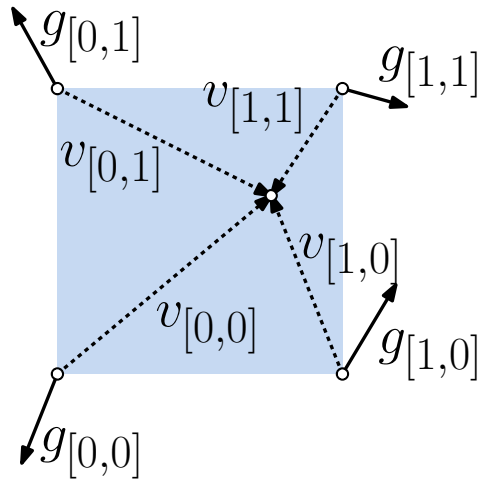
(a)

Random gradient vectors



(b)

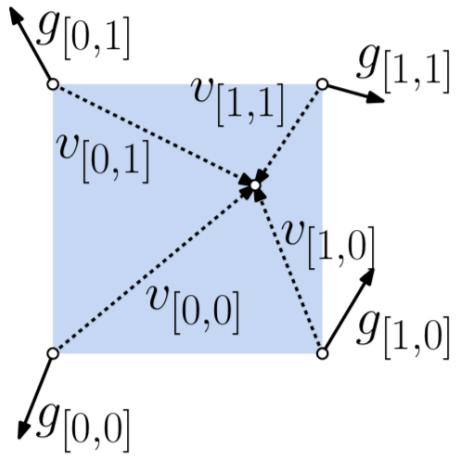
Smoothing/Interpolation



(c)

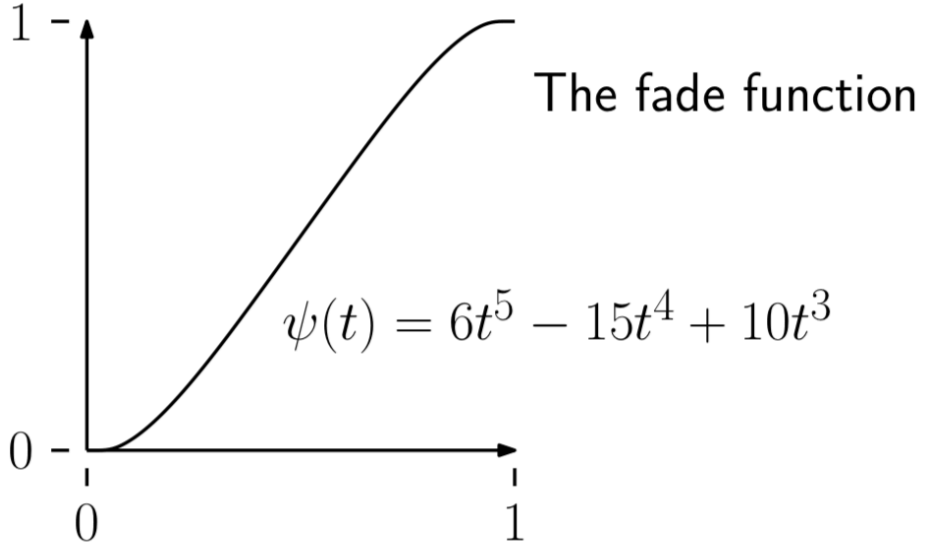
2D Perlin algorithm – gradient factor

$$\delta_{[0,0]} = (v_{[0,0]} \cdot g_{[0,0]}) \quad \text{and} \quad \delta_{[0,1]} = (v_{[0,1]} \cdot g_{[0,1]})$$
$$\delta_{[1,0]} = (v_{[1,0]} \cdot g_{[1,0]}) \quad \text{and} \quad \delta_{[1,1]} = (v_{[1,1]} \cdot g_{[1,1]}).$$



2D Perlin – fade function

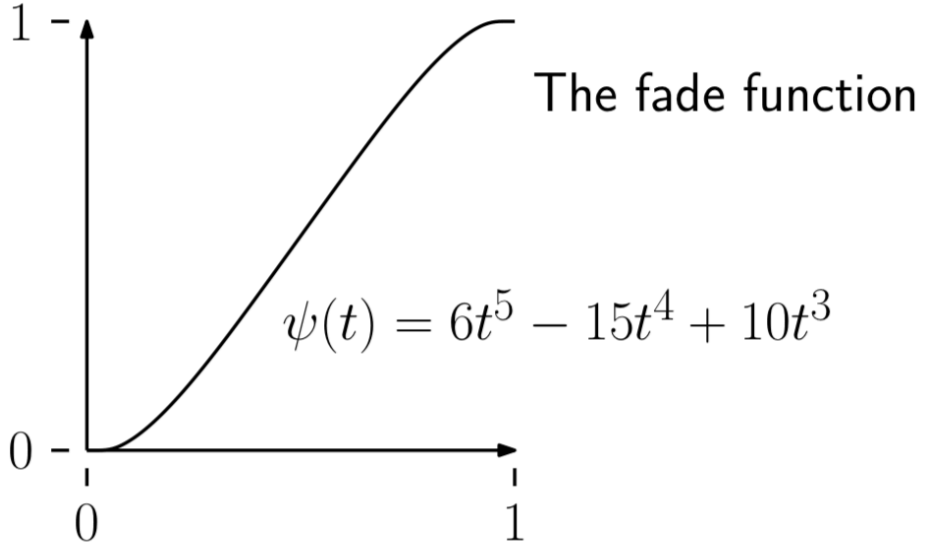
- $\psi(0) = 0$
- $\psi(1) = 1$
- $\psi'(0) = \psi'(1) = ??$



2D Perlin – fade function

- $\psi(0) = 0$
- $\psi(1) = 1$
- $\psi'(0) = \psi'(1) = ??$

- $\Psi(1) = \psi(s)\,\psi(t)$



2D Perlin – noise function

$$\text{noise}(x,y) \;=\; \Psi(1-x,1-y)\delta_{[0,0]} + \Psi(x,1-y)\delta_{[1,0]} + \Psi(1-x,y)\delta_{[0,1]} + \Psi(x,y)\delta_{[1,1]}$$

2D Perlin – noise function

$$\text{noise}(x,y) \;=\; \Psi(1-x,1-y)\delta_{[0,0]} + \Psi(x,1-y)\delta_{[1,0]} + \Psi(1-x,y)\delta_{[0,1]} + \Psi(x,y)\delta_{[1,1]}$$

$$\text{perlin}(x,y) \;=\; \sum_{i=0}^{k} p^i \cdot \text{noise}(2^i \cdot x, 2^i \cdot y)$$

- P is what parameter?

- 1. Metrics for best path on map
- 2. Navmesh process (R_D_P algorithm, triangulation)
- 3. Walkable terrain
- 4. Find paths on triangulated space
- 5. Configuration spaces
- 6. Quality of path
- 7. C-obstacles
- 8. Minkowski sums
- 9. Navmesh - grid, mulitresolution grid
- 10. Visibility graph
- 11. Medial axis
- 12. Randomized placement
- 13. Rapidly-expanded Random Trees (RRTs)
- 14. L-system plus turtle
- 15. Fractal dimension
- 16. Randomized and 3D L-systems
- 17. Particle systems
- 18. Flocking
- 19. Mandelbrot sets
- 20. Constructive solid geometry
- 21. Shading equation
- 22. Bump mapping
- 23. Polygonal meshes - basics, Euler's formula
- 24. DECL data structures
- 25. Perlin noise
- 26. A*
- 27. Admissible heuristic

Problem 3. (20 points) Consider the collection of shaded rectangular obstacles shown in the figure below, all contained within a large enclosing rectangle. Also, consider the triangular robot, whose reference point is located at a point s. (You may take s to be the origin.)

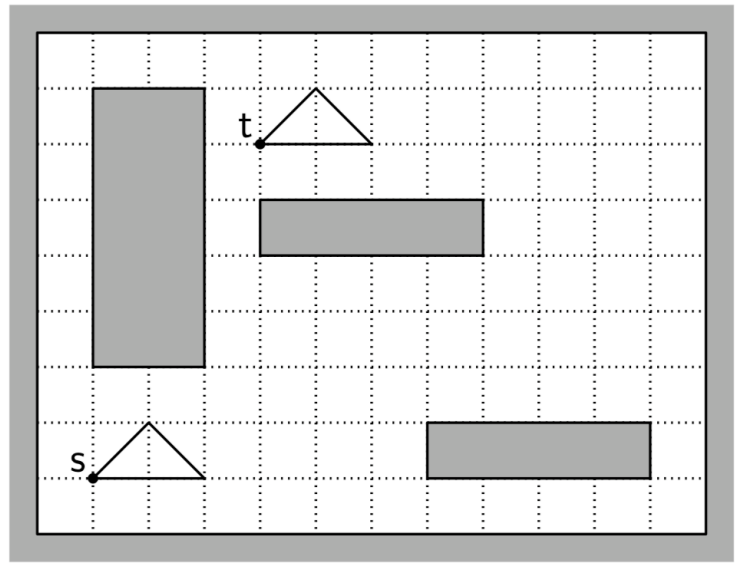


Figure 2: Problem 3.

(a) Draw the C-obstacles for the three rectangular obstacles, including the C-obstacle from region lying outside the large enclosing rectangle.

(b) Either draw an obstacle-avoiding path for the robot from s to t, or explain why it doesn't exist.