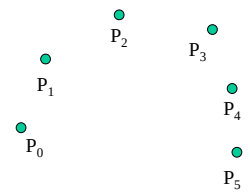
Lecture 16
B-Splines & Visibility

Michael Gleicher
October 27, 2005
Notes prepared as notes: Not projected for class

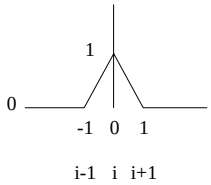
Consider B-Splines

- N points
  - General – any N
  - $P_0 \ldots P_{n-1}$
- WARNING: not same notation as book
- Consider linear interpolation
  - F(t) $t \in 0 \ldots n-1$
- $F(t) = \sum b_i(t)\, P_i(t)$



Linear B-Spline

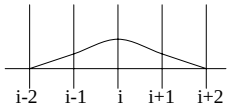
- Each blending function is a bump
- All the same (different ones are shifts)
- Active from i-1 to i+1
  - Over 2 spans, 3 integers
- In between 2 pts are active
  - One in each "phase"
- "before" t=0 and "after" n-1
  - Not enough points

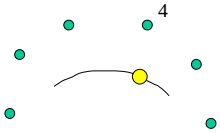


Warning:
This B-Spline is centered
Other notations start at 0

Cubic Blending Functions

- Active over 4 regions (d=3, k=d+1=4)
- At any time, one point in each phase
- Example t=4.5
  - Eval point 3 @ 1.5
  - Eval point 4 @ .5
  - Eval point 5 @ -.5
  - Eval point 6 @ -1.5
  - Each in a different part





How to make objects solid

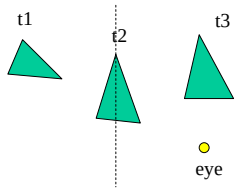
- So far, just curves (outlines of things)
- Can fill regions (polygons)
  - But how to get stuff in front to occlude stuff in back
- General categories
  - Re-think drawing
    - From eye (pixels) not objects
  - Analytically compute what can be seen
    - Hidden line drawing (hard)
  - Hidden Surface Removal

Painter's Algorithm

- Simplest hidden surface algorithm
- Draw farthest objects first
  - Nearer object cover further ones
- Problems
  - Cycles / intersections (no order possible)
    - Fix by splitting triangles
  - Need all triangles ahead of time
  - O(n log n) sort
  - Must resort for every view direction
- Depth Complexity (amount of time each pixels is drawn)

Binary Space Partitions

- Fancy data structure to help painters algorithm
- Stores order from any viewpoint
- A plane (one of the triangles) divides other triangles
- Things on same side as eye get drawn last



T2 divides into groups
T3 is on same side of eye

Using a BSP tree

- Recursively divide up triangles
- Traverse entire tree
  - Draw farther from eye subtree
  - Draw root
  - Draw closer to eye subtree
- Always O(n) to traverse
  - (since we explore all nodes)
  - No need to worry about it being balanced

Building a BSP tree

- Each triangle must divide other triangles
  - Cut triangles if need be (like painters alg)
- Goal in building tree: minimize cuts

Z-Buffer

- Throw memory at the problem
- A hardware visibility solution
  - Useful in software, but a real win for hardware
- For every pixel, store depth that pixel came from
- No object? Store ∞
- When you draw a pixel, only write the pixel if you pass the "z-test"

Things to notice about Z-Buffer

- Pretty much order independent
  - Same Z-values
  - Transparent objects
- Z-fighting
  - Objects have same Z-value, ordering is "random"
  - Bucketing (finite resolution) causes more things to be same
  - As things move, they may flip order
- Anti-Aliasing
  - Things done per-pixel, so sampling issues

Resolution of Z-Buffer

- Old days: big deal
  - Integer Z-buffers, limited resolution
- Future: floating point z-buffer
  - Still have resolution issues, not as bad
- Need to bucket things from near to far
  - Don't set near too near or far to far
- Non-linear nature of post-divide Z
  - Remember that perspective divide gives fn/z