CS50's
Introduction to Game Development

OpenCourseWare

Colton Ogden (https://www.linkedin.com/in/colton-ogden-0514029b/)
cogden@cs50.harvard.edu

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://orcid.org/0000-0001-5338-2522) (https://www.quora.com/profile/David-J-Malan) (https://www.reddit.com/user/davidjmalan)
(https://www.tiktok.com/@davidjmalan) (https://davidjmalan.t.me/)
(https://twitter.com/davidjmalan)

Lecture 11: Portal Problems

What is a Portal?

- A portal is a discontinuity in 3D space where the back face of a 2D rectangle is defined to be the front face of another 2D rectangle located elsewhere (but without actually moving any geometry or overlapping space in incomprehensible ways).
- They're essentially like doorways that can potentially take you far across the map.

Rendering: Texture vs Stencil Tradeoffs

- There are generally two main ways to render a portal view.
- Texture:
  - Separate textures per portal view (recursion gets big fast).
  - Use Painter's Algorithm (deepest portals first).
  - Can't effectively use antialiasing.

        - Simplest to implement when constrained to a depth of 1.
- Stencil:
    - Renders entire frame in the back buffer; no texture memory needed.
    - Start from the main view and recurse as necessary.
    - Homogenous visual quality.
    - Need to recurse after opaques, but before transparencies.

Rendering: Duplicate Models

- One interesting issue that was spotted early on was: how to handle the rendering of an object that is partly through a portal?
- To solve this, it was necessary to look up every single piece of rendering geometry used by the object, replicate it, and teleport it to the appropriate portal frame by frame.
- Of course, the result would be that now there would be two copies of the object sticking out through both sides of each portal.
- To fix this, the portals would need to be treated as “Clip Planes”, such that only one side of the object would be rendered (thus getting “clipped”).

Rendering: Recursion

- One thing that many people expected to see in Portal was infinite recursion when they looked at a portal through another portal.
- Of course, implementing infinite recursion would be impossible, but rendering the appearance of infinite recursion is definitely possible.
- To do this, recursion would be triggered only once for a portal within a portal, such that the first and second “layers” were legitimate.
- From then on, however, all further “layers” of the portal loops were fake - merely copies of the second layer.
    - This was a nice way of simulating the appearance of infinite portals, although it was not perfect.

Design: Prototyping in 2D

- Interestingly enough, before committing to spending several years working on a 3D Portal game, the developers first created a simpler 2D prototype to see whether the core mechanics of Portal were entertaining enough to base an entire 3D game around.

- This 2D prototype was opened to a small group of people as a sort of “soft opening” of portal - and the answer was yes!

Design: Levels and Game Mechanics

- Obviously, games are only fun when you know how to play them. This is why it’s very important to consider level design such that you can be sure that during each level, the player will have the necessary knowledge/experience to win.
- The best way to gather data for discovering what it is the player needs to know before playing is simply by asking others to play your game as you develop it.
- As you see how they interact with the game, this will lead you to discover problems you hadn’t accounted for or even player strategies that you hadn’t considered when creating a level.
- Ideally, as the player beats each level, they should be gaining knowledge that will become relevant during the next level.
- In Portal, the fun part of many levels was teaching the player the different ways in which they could interact with portals (e.g., portals teleport you to other portals, portals can be placed on walls, ceilings, and floors, portals are two-way, objects can also go through portals, etc.).
- Another important part of level design is knowing which features are worth implementing and which features are not.
- Oftentimes, an idea can sound very cool in theory but creates multiple complications in practice.
- In Portal, for example, there were many additional features that were considered as additions to the game but were ultimately scrapped since the cons outweighed the pros (e.g., sticky gel, portals within portals, double flings, etc.).

Combining Elements

- As discussed earlier, good level design ensures that within each level, the player is gaining knowledge/experience that will allow them to beat subsequent levels.
- A great way to do this is by having levels introduce the player to individual elements of the game, and slowly combine them as the player progresses.
- The result might be a very complicated level with many moving parts, but the player will not feel overwhelmed if they’ve previously had practice dealing with each of the elements of that level individually.

Physics: Volumes, Vectors, and Planes

- “Volumes” are essentially regions surrounding each portal that help keep track of objects in the vicinity, and there are two main volumes: green and yellow.
- The green volume is a fairly small region right in front of the portal that keeps track of objects that have a high probability of interacting with the portal.
- The yellow volume is a larger region (which contains the green volume) that keeps track of objects that are close to the green volume.
- Vectors and Planes refer to raycasting and portals.
- The physics in the game needs to implement behavior for what happens when a vector of some kind (e.g., a laser) collides with a portal.
- Namely, the vector needs to be split into two segments such that one segment is going into the portal and the other is coming out of the “destination” portal.