Ray Casting

What space to work in:

World or projection space?
World: objects are easy, viewport geometry needs calculation.
Projection: viewport is easy, objects must be transformed
trouble: projection space is 4D with homogeneous coordinates so we'll use world

Screen to world map

Pixel space:

A pixel (i,j) in $[0,w]\times[0,h]$

corresponds to a point (s_x, s_y) in projection space $[-1,1]\times[-1,1]$

$$(s_x, s_y) = \left(\frac{2}{w}(i+0.5)-1,\ \frac{2}{h}(j+0.5)-1\right)$$

A ray behind point (s_x, s_y) in projection space is determined by two points

$$\bar{A}=(s_x, s_y, 0,1)^T, \quad \bar{B}=(s_x, s_y, 1,1)^T$$

The world space points, A and B, that map to $\bar{A}$ and $\bar{B}$

$$\bar{A} = (PV)A$$
$$\bar{B} = (PV)B$$

can be solved for

$$A = (PV)^{-1}\bar{A} = V^{-1}P^{-1}\bar{A}$$
$$B = (PV)^{-1}\bar{B} = V^{-1}P^{-1}\bar{B}$$

Efficiency of screen to world map

We can do much better than inverting two points per ray, by writing points as linear combinations of **basis** vectors:

Let
$$\begin{aligned}\bar{O} &= (0,0,0,1)^T\\ \bar{X} &= (1,0,0,0)^T\\ \bar{Y} &= (0,1,0,0)^T\\ \bar{Z} &= (0,0,1,0)^T\end{aligned}$$
be a basis for points in projection space,

and
$$\begin{aligned}O &= V^{-1}P^{-1}\bar{O}\\ X &= V^{-1}P^{-1}\bar{X}\\ Y &= V^{-1}P^{-1}\bar{Y}\\ Z &= V^{-1}P^{-1}\bar{Z}\end{aligned}$$
be the corresponding basis in world coordinates

then

$$\bar{A}=(s_x, s_y, 0,1)^T = \bar{O} + s_x\bar{X} + s_y\bar{Y}$$
$$\bar{B}=(s_x, s_y, 1,1)^T = \bar{O} + s_x\bar{X} + s_y\bar{Y} + 1\bar{Z}$$

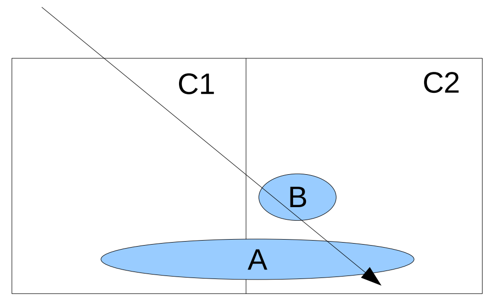
so

$$A = V^{-1}P^{-1}\bar{A} = O + sx\,X + sy\,Y$$
$$B = V^{-1}P^{-1}\bar{B} = O + sx\,X + sy\,Y + 1Z$$

Note that A and B are homogeneous coordinates,
so don't forget to do the homogeneous division

Problem with spatial decomposition enhancements:

If the spatial decomposition method
- does not **split** objects across boundaries, then
- objects can be in more than one cell
- objects can flow outside a cell



Wrong:
- Ray R hits cell C1 first
- C1 contains A
- R intersects A
- Conclude R intersect A first

Right:
- Ray R hits cell C1 first
- C1 contains A
- R intersects A (but not in current cell C1)
- Record Intersection(R,A), but continue

- Ray R hits cell C2 next
- C2 contains A and B
- Intersection(R,B) is found
- Intersection(R,A) is known from previous cell
- Front most of intersections is Intersection(R,B)

Rules:
- In a cell, for each object Ob contained/touching a cell:
  - lookup previously calculated Intersection(R,Ob),
  - or if not found, calculate the Intersection(R,OB)
- If one or more intersection are **in the current cell**:
  - choose the front most
- If any intersections are **outside the current cell**:
  - record intersection point for later use,
  - and continue along R to next cell.