Graphics in Games

level of detail in models
lighting
colors
ray casting

terrain maps
texture maps
shadows

take 4390–4391/4451!

How have graphics influenced the development of games?

What would you do with substantially more graphics computing power?

How have graphics influenced the development of games?

- vertical walls/horizontal floors
- texture mapping vs modeling detail
- fog
- games set indoors

What would you do with substantially more graphics computing power?

- lighting effects––reflections, shadows
- more texture memory
- more animation (pipeline latency)
- more simulation or AI from CPU

Level of Detail

- using simplified versions where detail is not needed
- used throughout the system
  - polygon meshes
  - textures (procedural and not)
  - animation (procedural and not)
- where does LOD info come from?
- how/when to swap models?
  - less important if details are truly minor but pops may attract attention

Level of Detail: Modeling

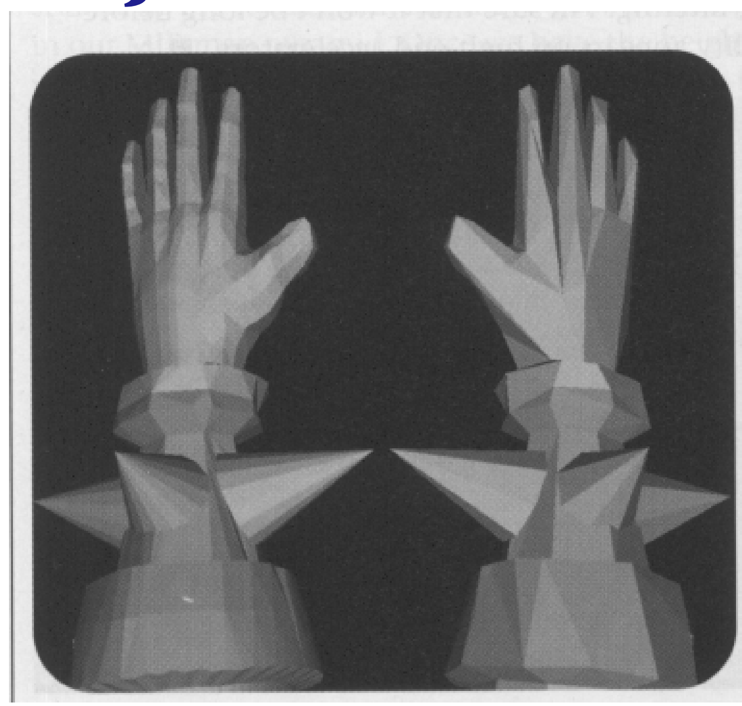
by hand or automatic?

why artists will do better:

- knowledge about the model
  - facial features, silhouette

why use automatic meshing?

- dynamically changing objects
- cpu vs. artist time

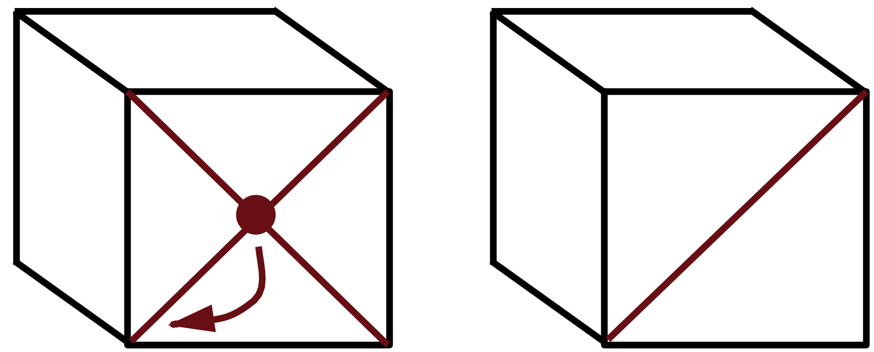


Art of Low Polygon Modeling

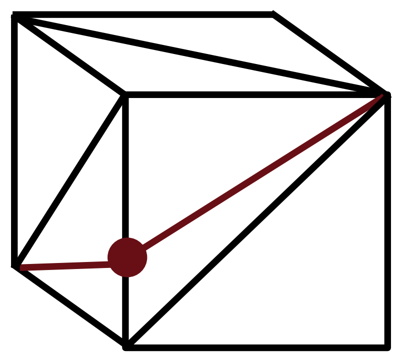
- know your limitations
  - target face count
    - Quake II 600 faces/character
  - engine depends on vertices or faces?
- know what matters
  - how will model be seen?
    - back or front? near or far?
    - alone or in groups?
- properties of model
  - closed model?
  - one model or articulated?
  - organic vs. non–organic

Techniques for Low Poly Models

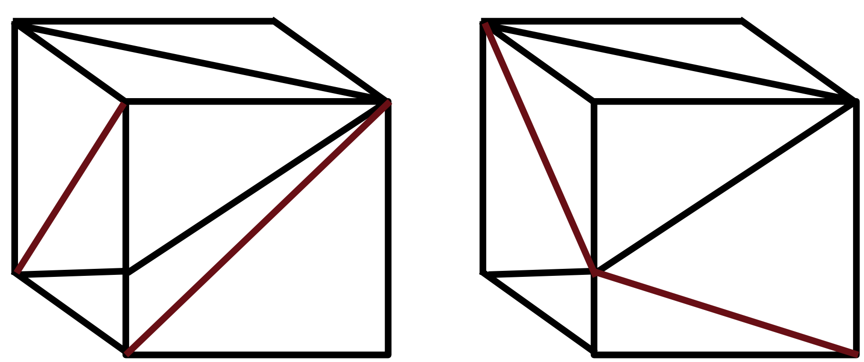
vertex merge



edge division



edge turn

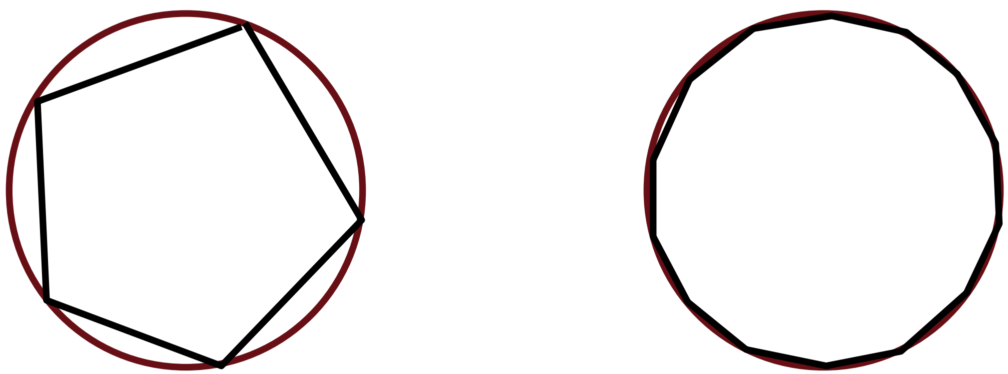


Automatic Meshing Techniques

off–line (Siggraph Community)

on–line: heuristics for vertex deletion

on–line: parametric surfaces



Progressive Meshing

selecting which edge to collapse next

- small details go first
- nearly co–planar surfaces need fewer polygons than areas of high curvature

$$\mathrm{cost}(u,v) = \|u-v\| \max_{f \varepsilon T_u} \left(\min_{n \varepsilon T_{uv}} (1 - f_n \cdot n_n)/2\right)$$

where T_u is set of triangles that contain u, T_{uv} is set of triangles that contain uv

Progressive Meshing

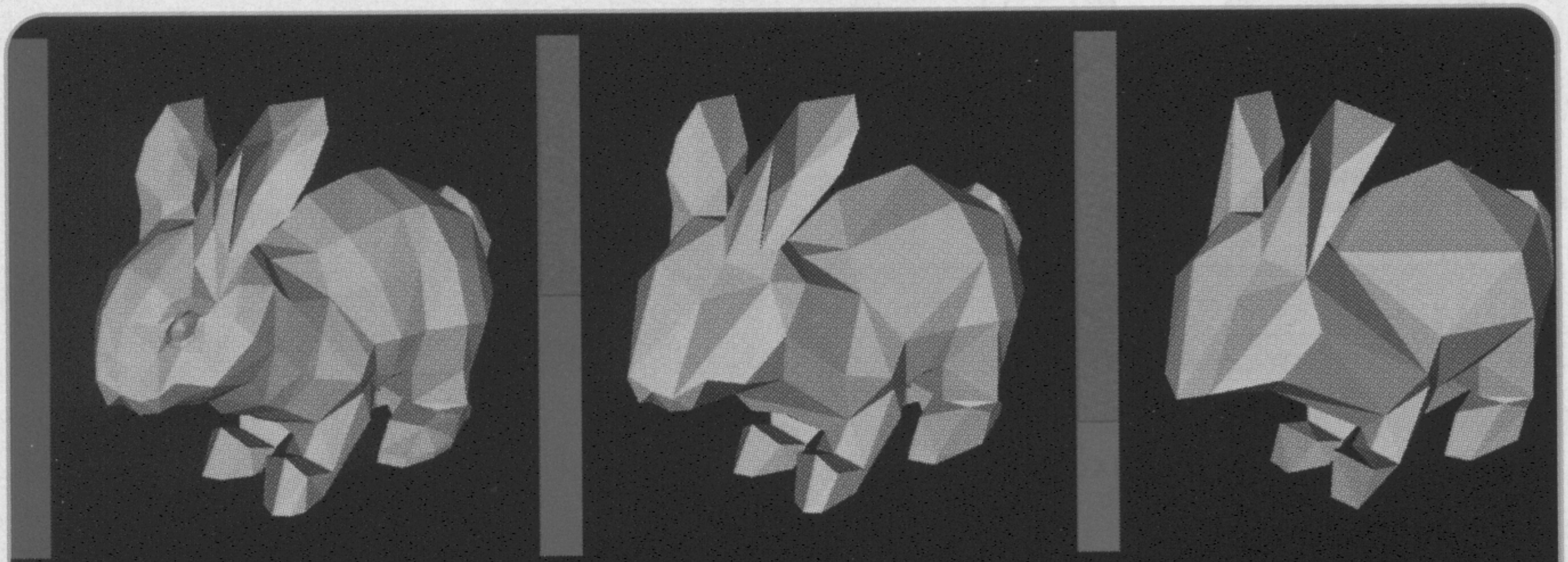


FIGURE 5. *Bunny model at (left to right) 453, 200, and 100 vertices.*

Lighting

- video games are different than stills or even animations: viewpoint, object motion

Goals

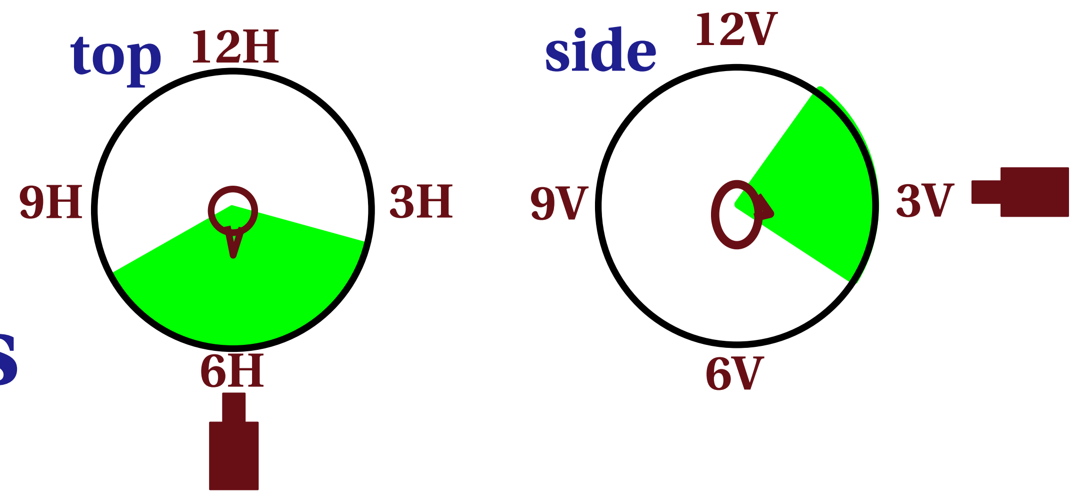
- direct viewer's attention
- emphasize depth and separation
- reveal texture, form
- create mood
- provide exposure and balance

Properties of Lights

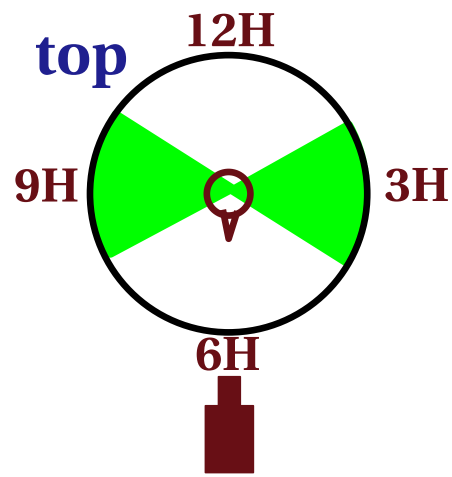
- quality: hard or soft
- intensity: want objects to differ in brightness for separation and depth
- color and pattern: glow from sunset, grid from bars
- direction: frontal, edge/side, back

Direction

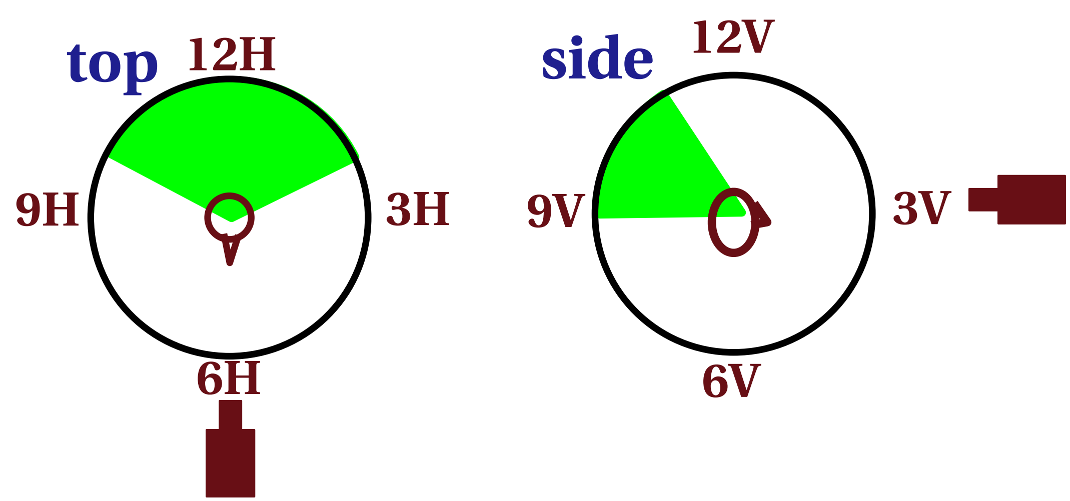
frontal: key light strongest, shadows



edge/side: contours, texture



back: separate from background



Key light

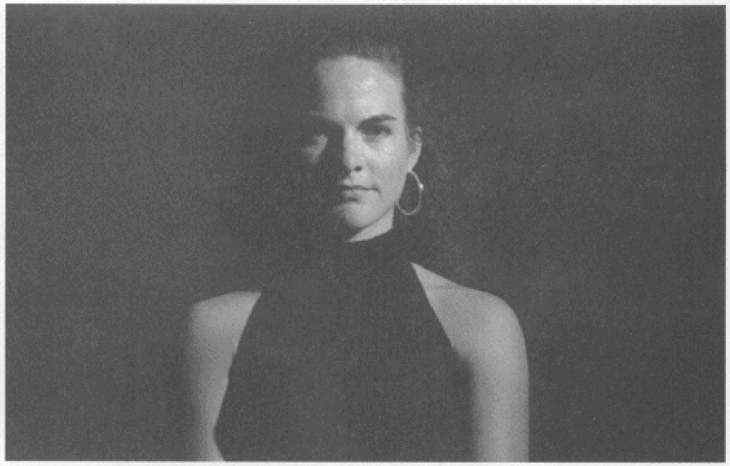
- casts shadows
- chief light source
- diagonally from front, high

Fill light

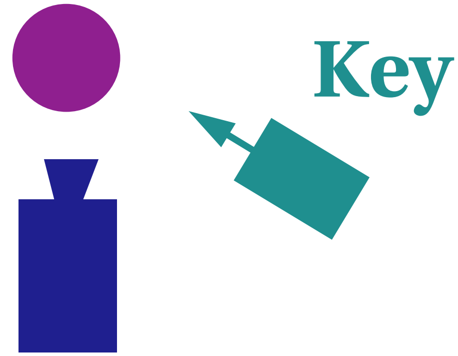
- soften shadows
- lower intensity
- positioned lower

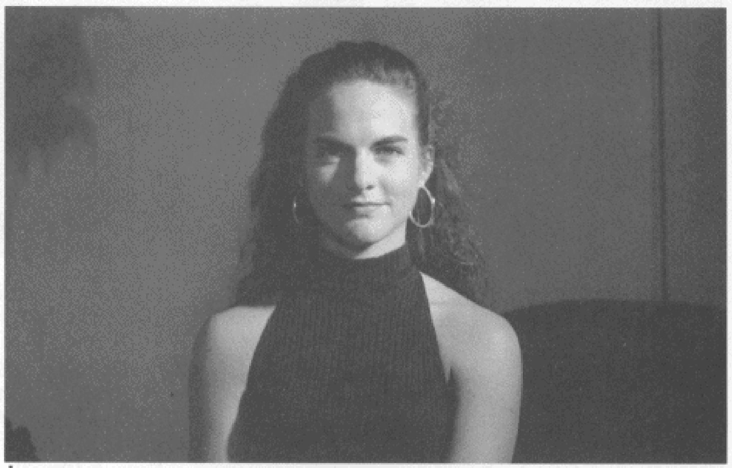
Back light

- separate figure from background (3d)
- mid–level intensity
- above figure

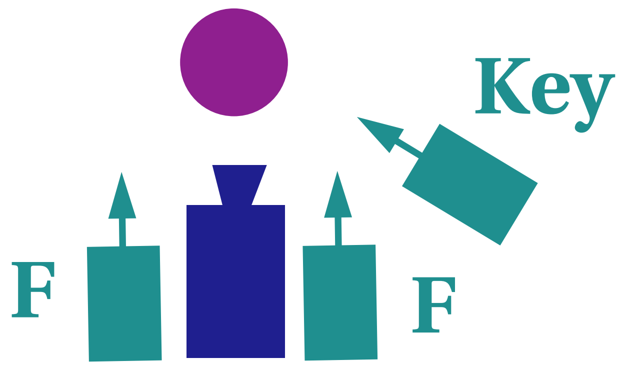


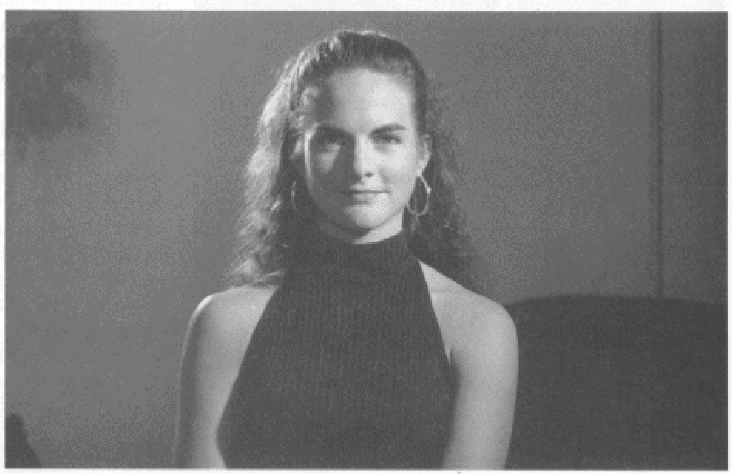
a



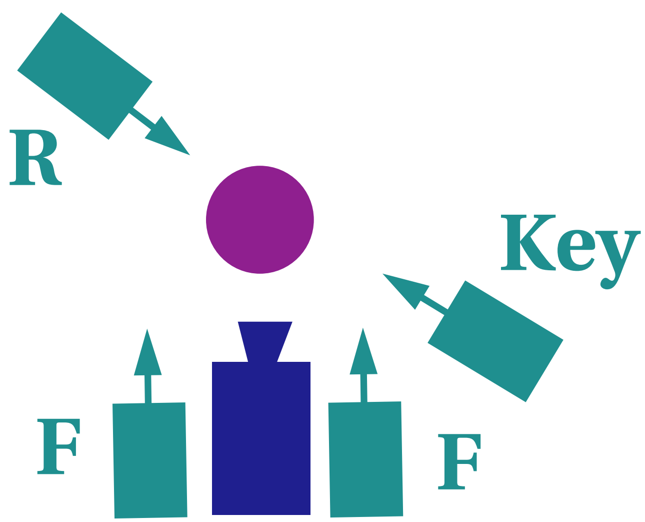


b





c



Color Theory

use color palette to set mood

color temperature

warm: red, red–orange, yellow, yellow–green

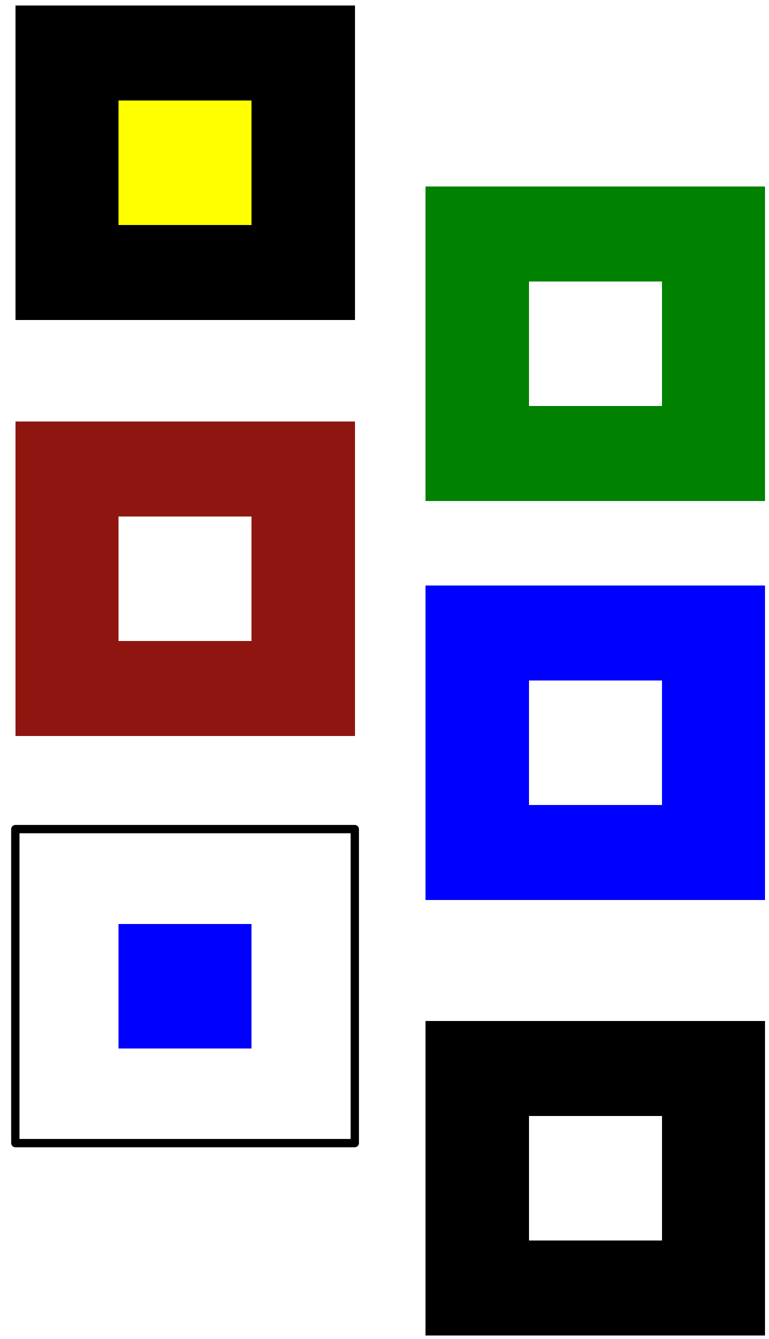
cold: violet, blue, green, green–yellow, blue–green

weight: darker–>heavier

depth: grey–> more distant

visibility:

black/yellow
green/white
red/white
blue/white
white/blue
black/white



Color Schemes that Work

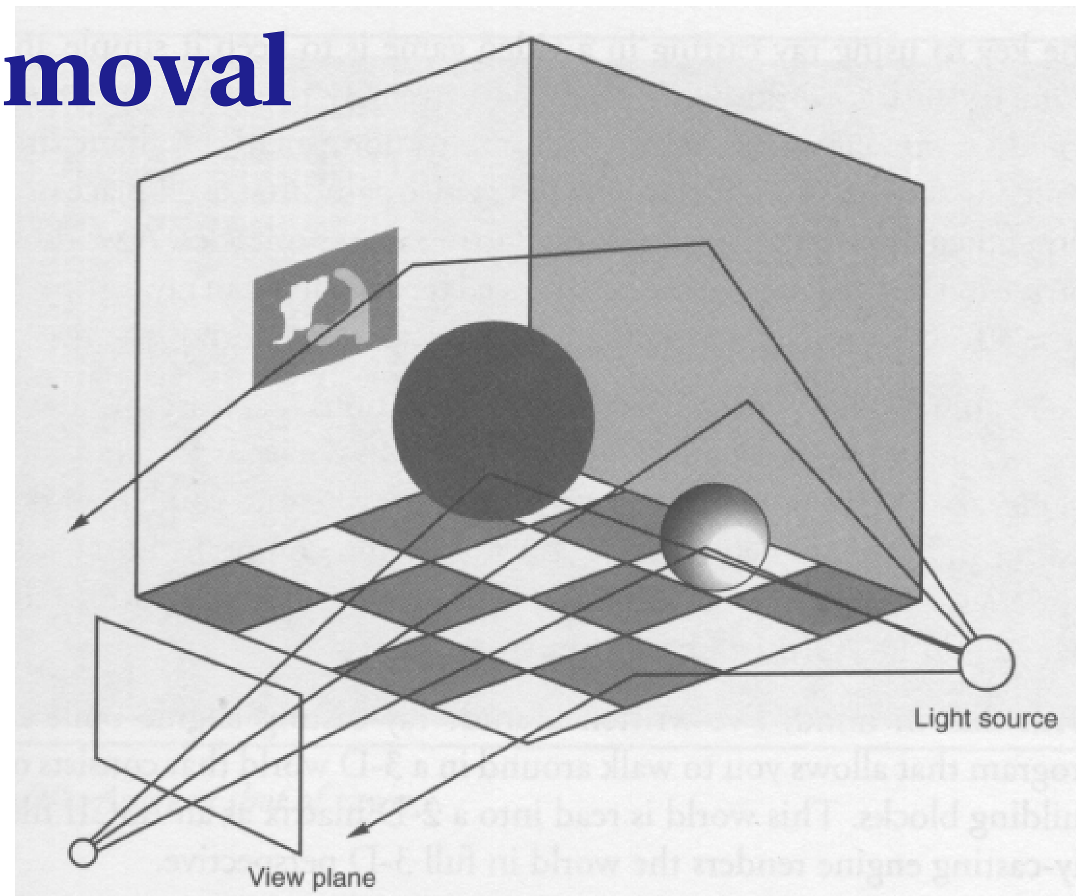
- monochromatic
- just primary colors
- all warm or all cool
- contrast of warm and cool
- color and its complement

Raycasting

- Wolfenstein 3D in 1992
- subclass of ray tracing

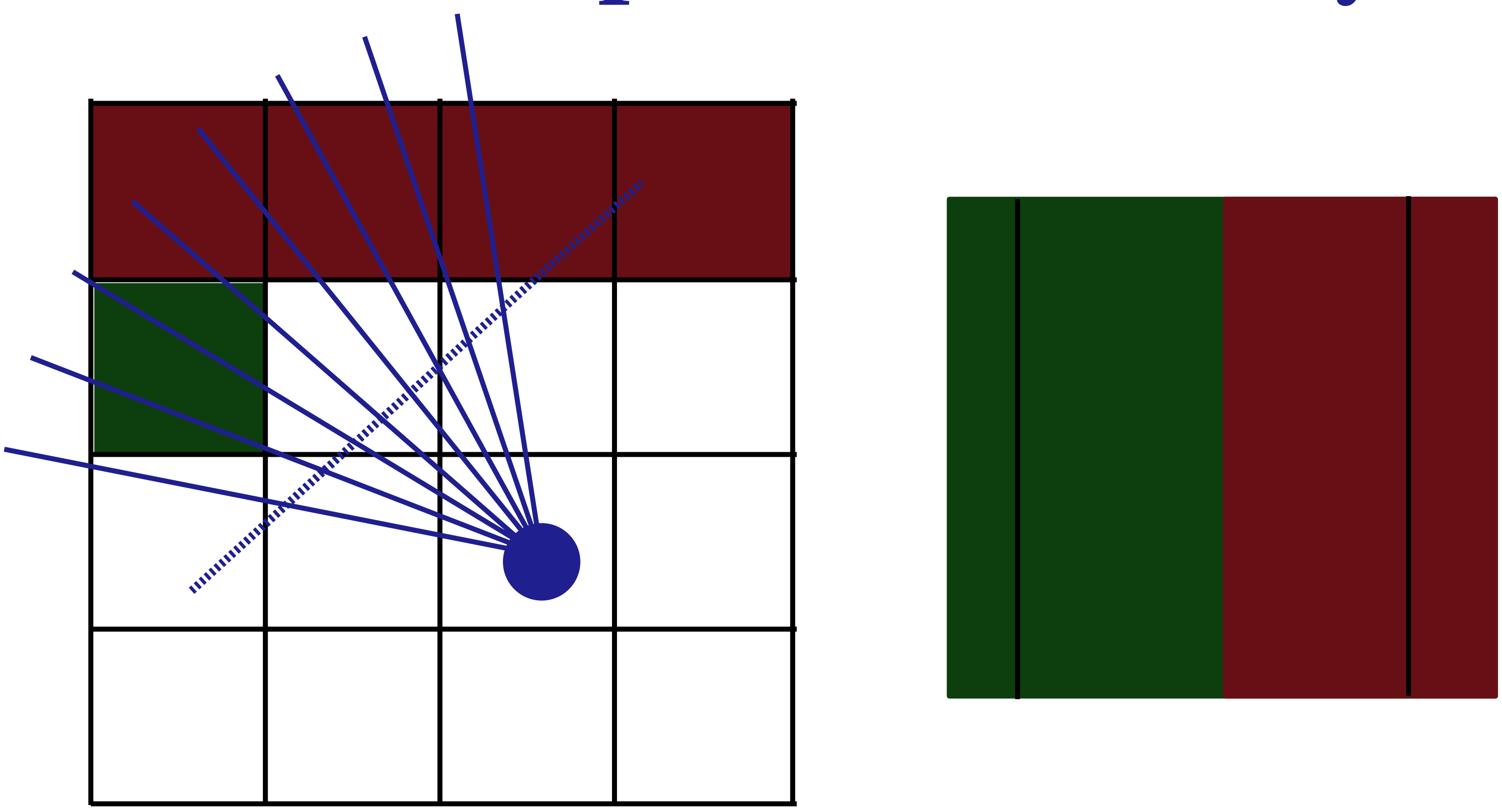
Raytracing

- hidden surface removal
- transparency
- reflections
- refractions
- ambient lighting
- shadows



Raycasting

grid world plane
number of rays –> horizontal resolution
        + subsampling?
tables of slopes for efficiency

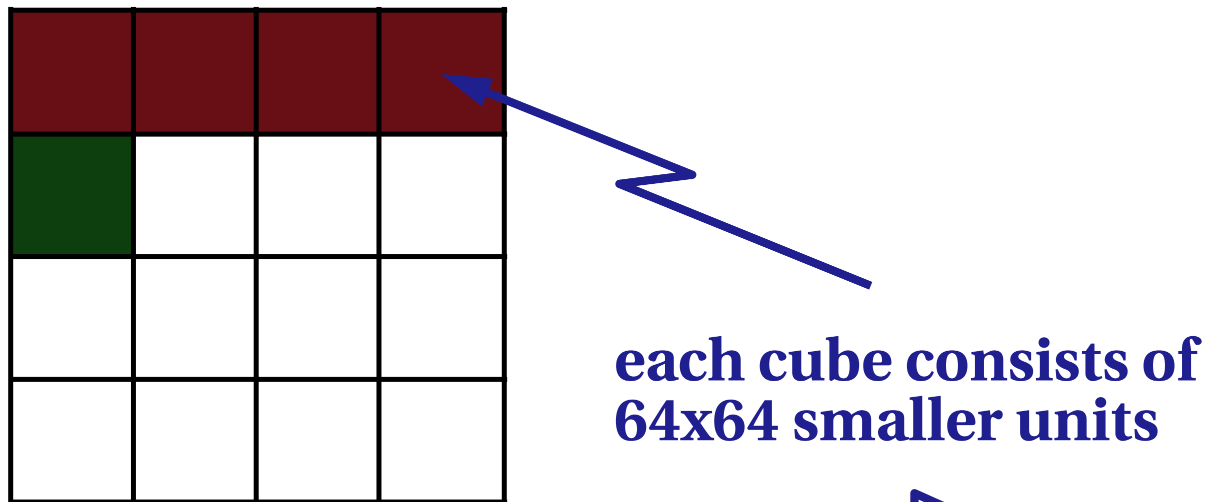


World

walls at 90^{o} wrt to floor

walls made of uniform cubes

floor flat

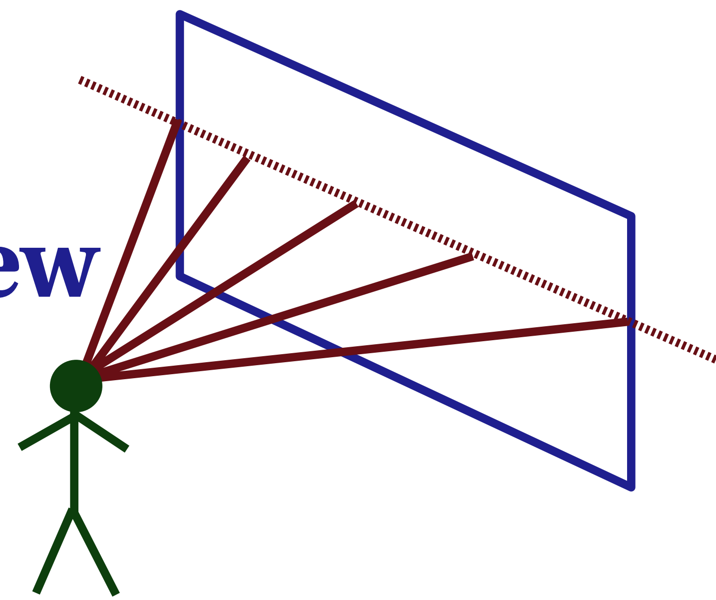


Viewer

player's height, field of view

x,y position of player

facing direction (yaw)



FOV = 60
screen size = 320

Finding Walls

ray = viewing angle –30

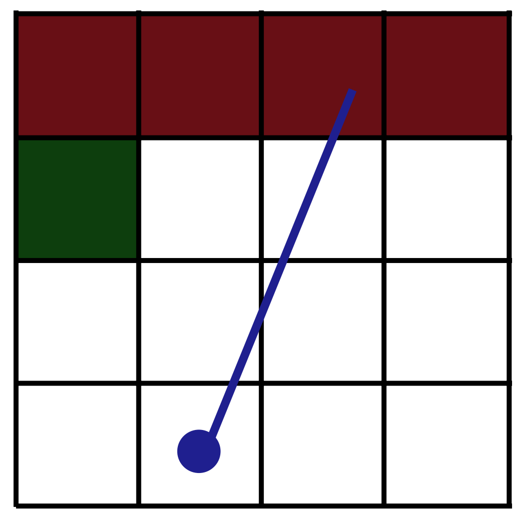
for (col = 0; col++, ray += 60/320, col<320)

follow ray until hit wall

record distance to wall

Finding Intersections

find intersection
points with the grid



fixed number of
ray angles: 360/(60/320)
use a table for the slope

Horizontal Intersections

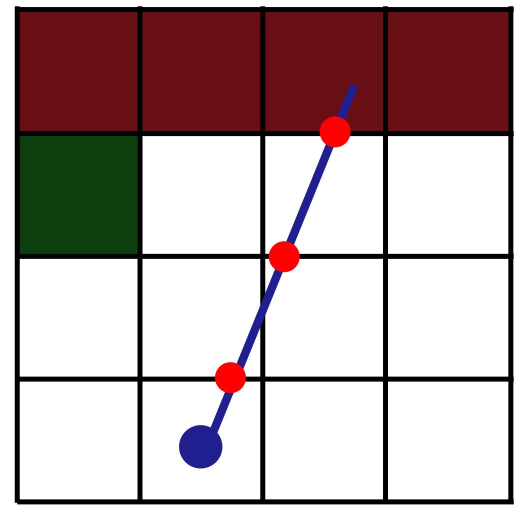
find first intersection

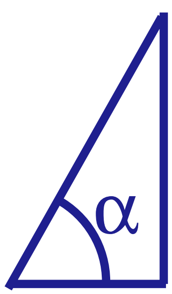
check grid (wall or !wall)

if wall compute distance

if !wall

- find next intersection
- x’ = x+xa, y’ = y+ya
- xa = table lookup
- ya = grid height

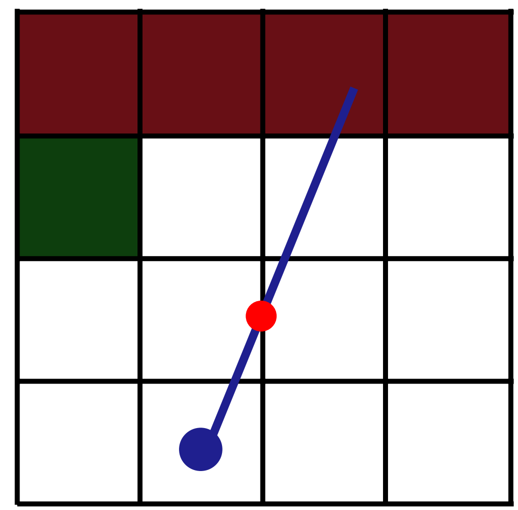




ya = 64

xa = 64/tanα

xa?



Vertical intersections are similar––look up ya, xa is grid width

Finding Distance

$d = \mathrm{sqrt}((px - dx)^2 + (py - dy)^2)$

$d = \mathrm{abs}(px - dx)\ /\ \cos(\alpha)$

$d = \mathrm{abs}(py - dy)\ /\ \sin(\alpha)$

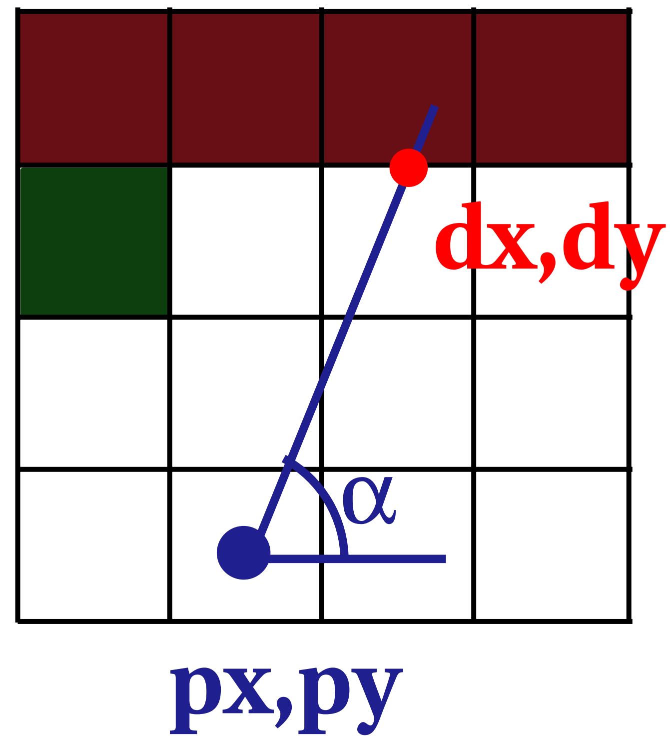


table look–up for cos, sin
finite number of values for α

Improvements

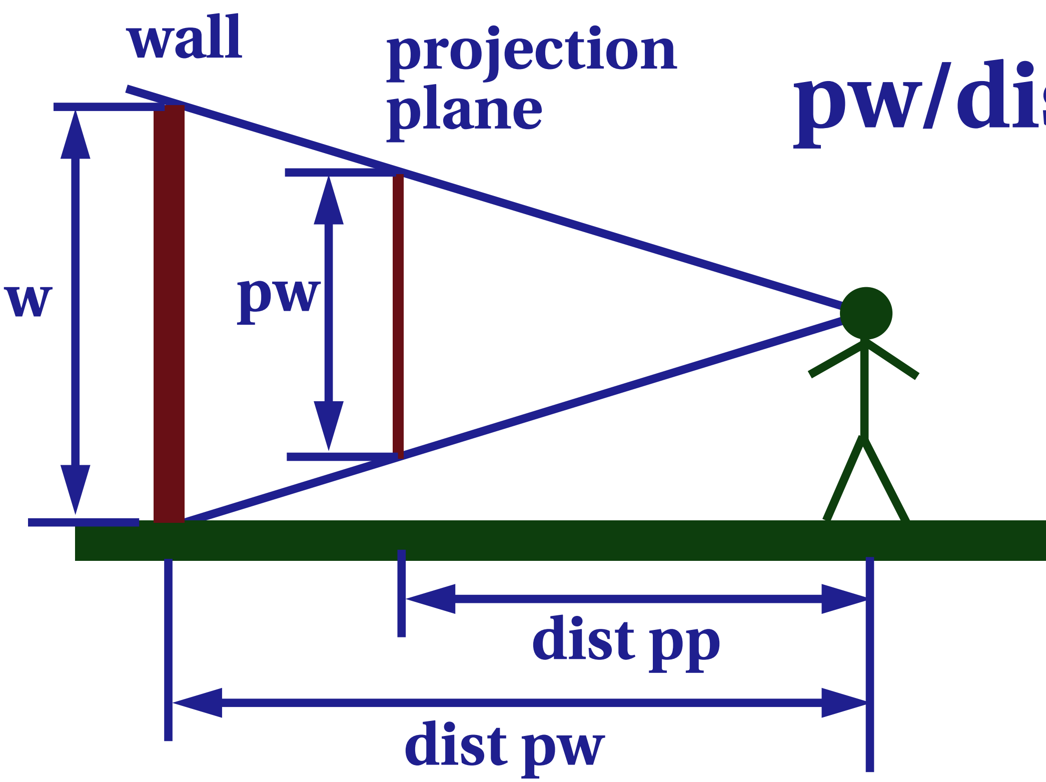
doors and windows

45^o walls

platforms and ramps

Drawing Walls

find height of projected wall slice



pw/dist pp = w/dist pw