Interactive Computer Graphics

- Lecture 13:
- Constructive Solid Geometry (CSG)

Constructive Solid Geometry (CSG)

- Real and virtual objects can be represented by
  - solid models such as spheres, cylinders and cones
  - surface models such as triangles, quads and polygons
- Surface models can be rendered either by
  - object-order rendering
  - image-order rendering (i.e. ray tracing)
- Solid models can only be rendered by ray tracing
- Solid models are commonly used to describe man-made shapes
  - computer aided design
  - computer assisted manufacturing

Constructive Solid Geometry (CSG)

- Constructive solid models can consists of primitive shapes such as
  - sphere
  - cylinder
  - cone
  - pyramid
  - cube
  - box

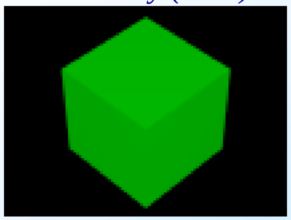
  (sphere, cylinder, cone: } Implicit representation)
- Constructive solid models cannot consist of half-spaces such as
  - points
  - lines
  - planes

Constructive Solid Geometry (CSG)

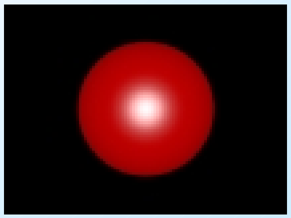
- CSG combines solid objects by using three (four) different boolean operations
  - intersection ($\cap$)
  - union (+)
  - minus (–)
  - (complement)
- In theory the minus operation can be replaced by a complement and intersection operation
- In practice the minus operation is often more intuitive as it corresponds to removing a solid volume

Constructive Solid Geometry (CSG)

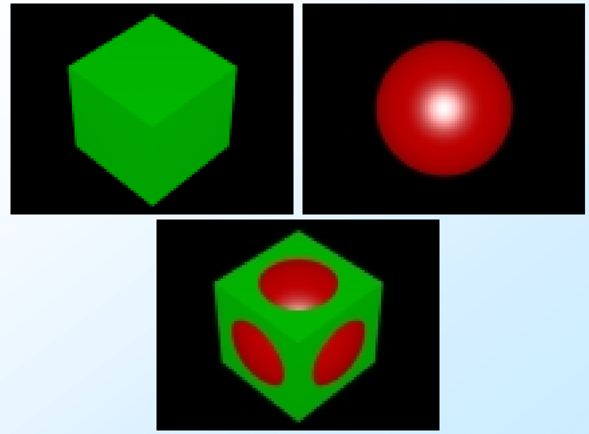
- Box



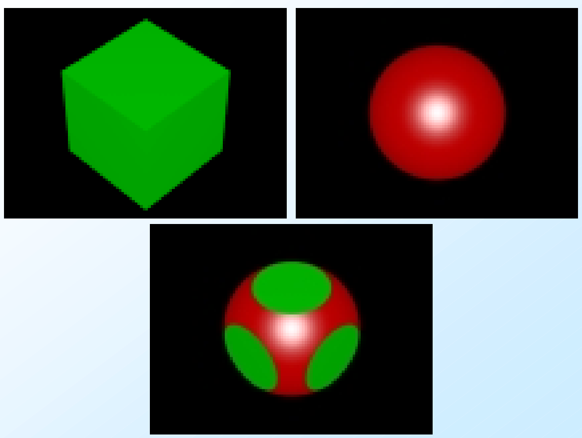
- Sphere



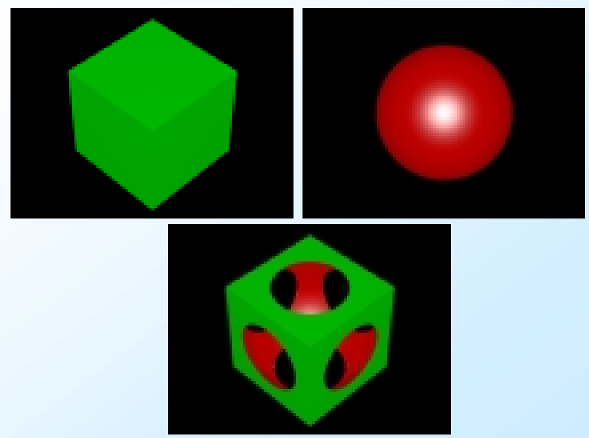
Constructive Solid Geometry (CSG): Union



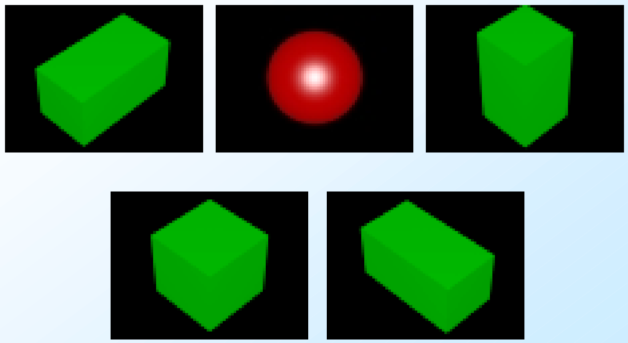
Constructive Solid Geometry (CSG): Intersection



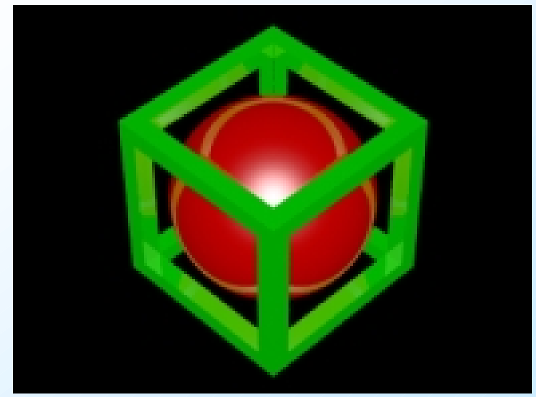
Constructive Solid Geometry (CSG): Minus



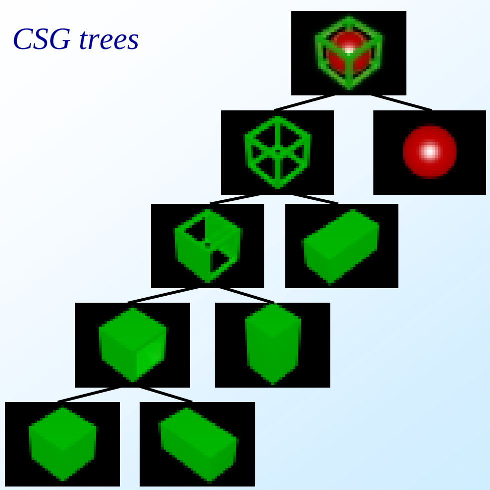
CSG trees



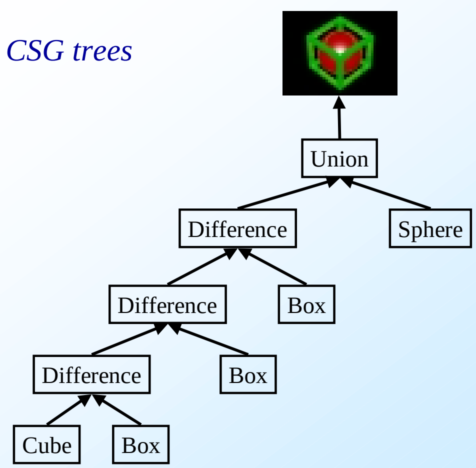
CSG trees



CSG trees

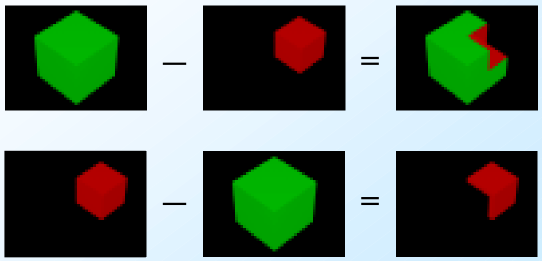


CSG trees



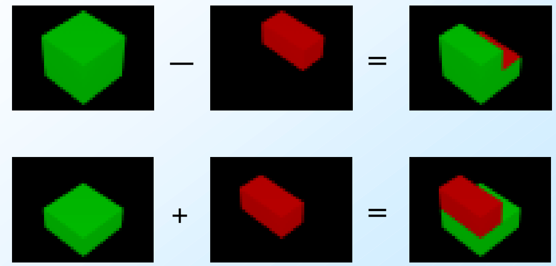
CSG trees

- CSG operations are not commutative:

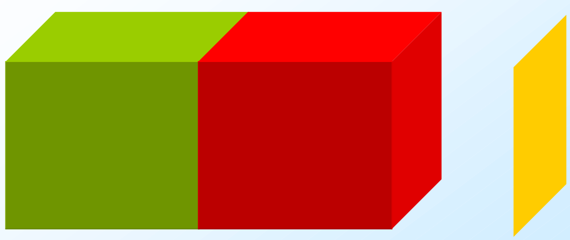


CSG trees

- CSG operations are not unique:

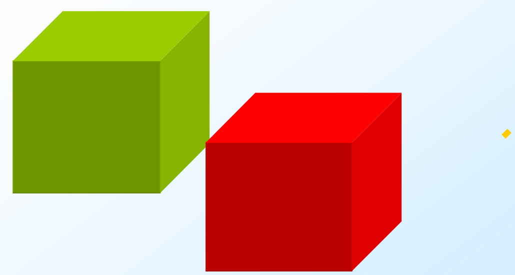


CSG trees: Problems



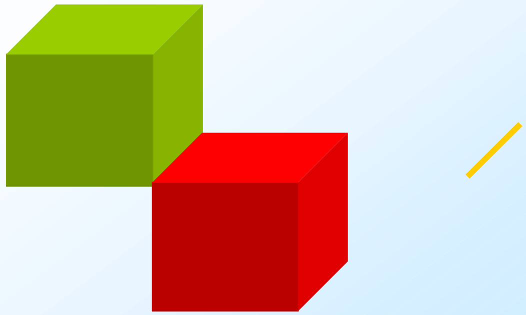
Cube A ∩ Cube B = Plane

CSG trees: Problems



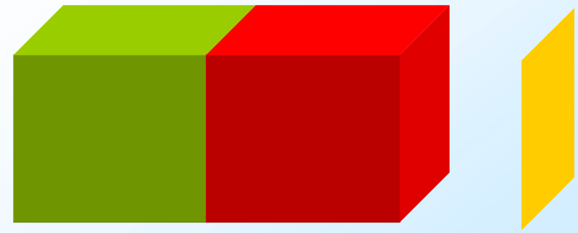
Cube A ∩ Cube B = Point

CSG trees: Problems



Cube A ∩ Cube B = Line

CSG trees: Problems



Cube A ∩ Cube B = Plane

CSG: Interior, Exterior and Closure

- a point **p** is an interior point of a solid **s** if there exists a radius *r* such that the open ball with center **p** and radius *r* is contained in the solid **s**. The set of all interior points of solid **s** is the interior of **s**, written as **int**(**s**). Based on this definition, the interior of an open ball is the open ball itself.
- a point **q** is an exterior point of a solid **s** if there exists a radius *r* such that the open ball with center **q** and radius *r* is not contained in **s**. The set of all exterior points of solid **s** is the exterior of solid **s**, written as **ext**(**s**).

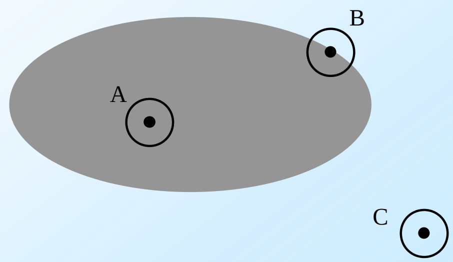
CSG: Interior, Exterior and Closure

- all points that are not in the interior nor in the exterior of a solid **s** are the boundary of solid **s**. The boundary of **s** is written as **b**(**s**). Therefore, the union of interior, exterior and boundary of a solid is the whole space.
- the closure of a solid s is defined to be the union of **s**'s interior and boundary, written as **closure**(**s**). Or, equivalently, the closure of solid **s** is all points that are not in the exterior of **s**.

CSG: Interior, Exterior and Closure

- Definition of an open ball:

$$(x_0 - x)^2 + (y_0 - y)^2 + (z_0 - z)^2 < r^2$$



CSG: Interior, Exterior and Closure

- Consider a sphere:

$$x^2 + y^2 + z^2 = r^2$$

- The interior of a sphere is

$$x^2 + y^2 + z^2 < r^2$$

- The closure of a sphere is

$$x^2 + y^2 + z^2 \leq r^2$$

CSG: Interior, Exterior and Closure

- The exterior of a sphere is

$$x^2 + y^2 + z^2 > r^2$$

- A solid is a three-dimensional object
  - The interior of a solid is a three-dimensional object
  - The boundary of a solid is a two-dimensional surface

CSG: Interior, Exterior and Closure

- To eliminate these lower dimensional branches, the three set operations are regularized:
  - Compute the result as usual and lower dimensional components may be generated.
  - Compute the interior of the result. This step would remove all lower dimensional components. The result is a solid without its boundary.
  - Compute the closure of the result obtained in the above step. This would add the boundary back.

CSG: Interior, Exterior and Closure

- Let +, ∩ and – be the regularized set union, intersection and difference operators. Let A and B be two solids. Then, A + B, A ∩ B and A – B can be defined mathematically based on the above description:
- A + B = **closure**(**int**(the set union of A and B)
- A ∩ B = **closure**(**int**(the set intersection of A and B)
- A – B = **closure**(**int**(the set difference of A and B)

Ray tracing CSG trees

- CSG trees must be rendered by ray tracing
- CSG trees must be traversed in a depth first manner
  - traversal starts at the leaf nodes
  - traversal of each node yields a list of line segments of the ray that pass through the solid object
  - list of lines segments is passed to parent node and processed accordingly

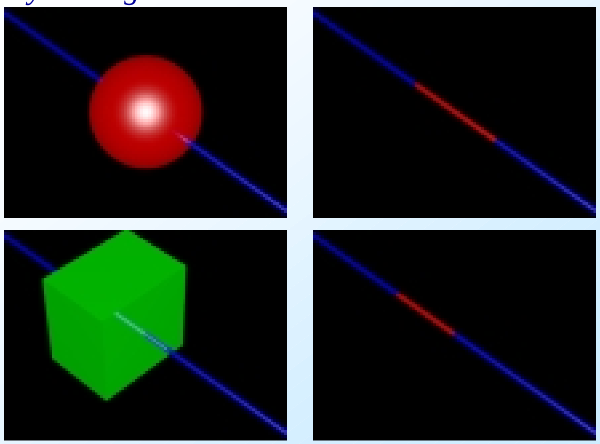
Ray tracing CSG trees

- Each list of line segments (or spans) may be characterized by the alpha values representing the intersection points of the corresponding ray equation:

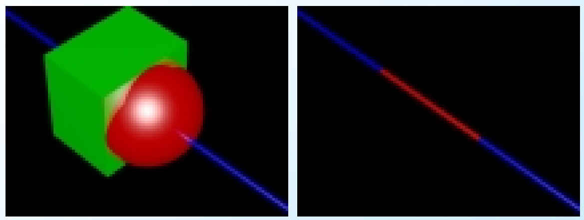
$$(\mu_1, \mu_2, \ldots, \mu_n)$$

- Each list of line segments will either contain
  - an odd number of intersection points (the viewpoint is inside the solid object)
  - an even number of intersection points (the viewpoint is outside the solid object)
  - an empty list of intersection points

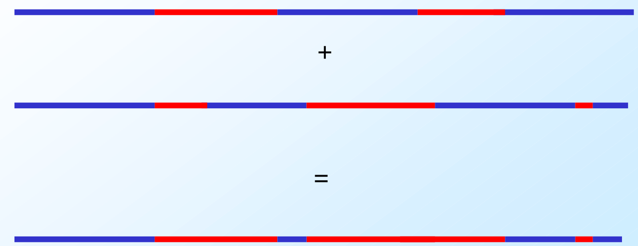
Ray tracing CSG trees



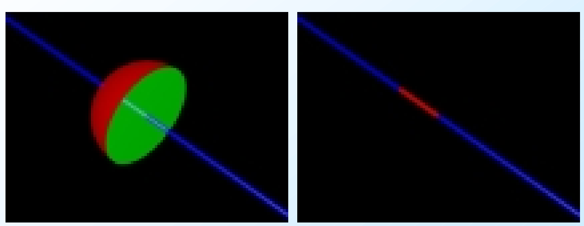
Ray tracing CSG trees: Union



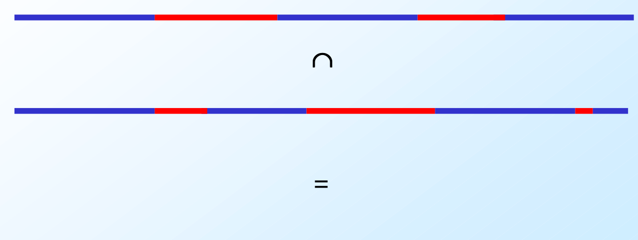
Ray tracing CSG trees: Union



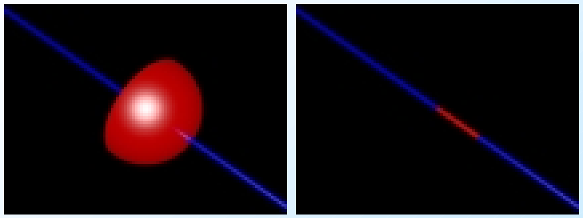
Ray tracing CSG trees: Intersections



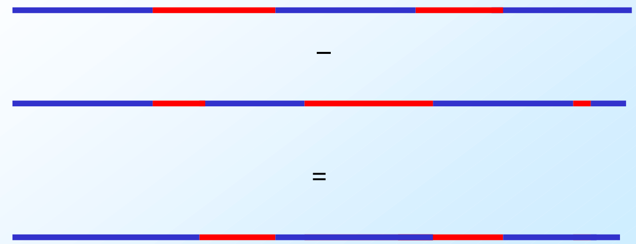
Ray tracing CSG trees: Intersections



Ray tracing CSG trees: Minus



Ray tracing CSG trees: Minus

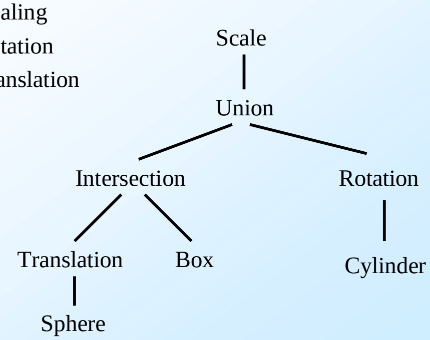


Ray tracing CSG trees

- CSG trees can be pruned during ray tracing:
  - if the left or right subtree of an intersection operation returns an empty list, then the other subtreed need not be processed.
  - if the left subtree of a minus operation returns an empty list, then the right subtree need not be processed.
- CSG trees can use bounding boxes/spheres to speed up rendering:
  - each primitive that does not belong the currently processed bounding volume may be represented by an empty intersection list

Extending CSG trees

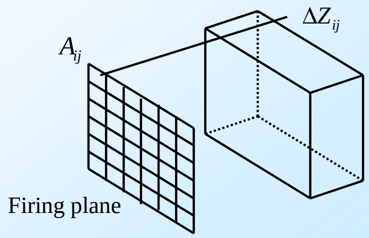
- Adding transformations as primitive operations:
  - scaling
  - rotation
  - translation



Calculating properties of CSG trees

- Ray tracing can be used to approximate the physical properties of objects including
  - volume
  - mass

  by firing a set of parallel rays from a firing plane



Calculating properties of CSG trees

- Volume:

$$V \approx \sum V_{ij}$$

  where

$$V_{ij} = A_{ij} \Delta Z_{ij}$$

  - A_{ij} is the area in the firing plane
  - ΔZ_{ij} is the distance between two intersection points along the firing ray
- Mass:

$$M \approx \sum M_{ij}$$

$$M_{ij} = A_{ij} \Delta Z_{ij} \rho_{ij}$$

Summary

- CSG representations are compact and efficient:
  - unevaluated models can be modified easily and efficiently
  - unevaluated models must be processed (by evaluating the CSG tree before further operations can be carried out
- CSG representations are valid, i.e. they create solid models
- CSG representations are accurate, i.e. they can represent curved surfaces
- CSG representations are not unique
- CSG representations are slow to render, i.e. they must be rendered using ray tracing
- CSG representations can be converted to polygonal representations and then rendered