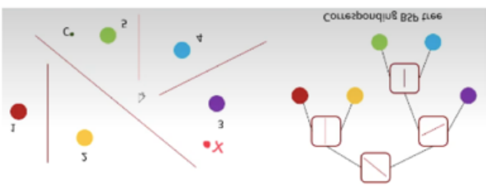
notes 2

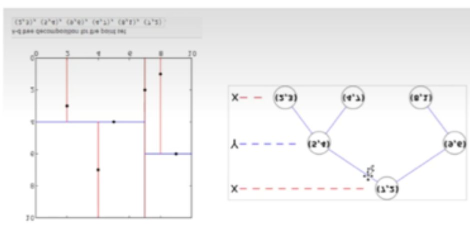
↳ k-D tree   cells aspect ratio can be high

- a scene is divided into 2 convex sets by axis-aligned hyperplanes (space subdivision)
- a special case of BSP   (it will be balanced if split according to median) ↳ log(n) depth
- each node is a k-dimensional point



for point C → 1-5-3-4-2
X → 1-5-2-4-3

Binary Space Partitioning (BSP)



- A 2-D example (k = 2):

K-D Tree

↳ bounding volume hierarchy (BVH)

- partitions objects into disjoint groups (object subdivision)
- overlap can occur between bounding boxes (volumes)
- hierarchy → parents are covering children boxes
- significant speed up in ray tracing

texture mapping

↳ no fixed d, s, β reflectance values for textures, their values varies

↳ in image, we take each point (u,v), calculate its mapping in the texture image mapping (i,j)

↳ mapping must be 3D → 2D
- triangles: x,y,z → u,v using barycentric coordinates (β,γ)
- spheres: x,y,z → u,v using angles

step 1 = associate $(u,v) \in [0,1]\times[0,1]$ coordinates to texture image

step 2 = parameterise the surface using two coordinates

step 3 = compute a (u,v) val for every surface point

step 4 = find the texture image coordinates (i,j) at the given (u,v) coordinates

step 5 = choose the texel color using a suitable interpolation strategy

↳ nearest neighbor = fetch texel at nearest coordinate (faster) (rounding)

↳ bilinear interpolation = average four closest neighbor (weighted) (smoother)

} texture values coordinates

↳ after getting the texture color → replace shading color, replace kd component ....

texture mapping triangles: associate (u,v) coordinates for each vertex

$u(\beta,\gamma) = u_a + \beta(u_b - u_a) + \gamma(u_c - u_a)$

$v(\beta,\gamma) = v_a + \beta(v_b - v_a) + \gamma(v_c - v_a)$

} barycentric coordinates

(i,j) = pixel value to be mapped (texture)

tiling: process of repeating a texture inside of rectangle (a square)

↳ texture coordinates are not limited to [0,1]

texture atlases: combines textures of many objects into a single image

procedural textures: generating textures procedurally for each surface point, instead of using an image as the texture to be mapped

↳ to look natural/realistic random patterns (ex: perlin noise technique)