# CG Practice 10

COLLEGE OF COMPUTING

HANYANG ERICA CAMPUS

Q YOUN HONG (홍규연)

# Spatial Data Structures

# Spatial Data Structure



Spatial data structure (공간자료구조)의 중요성:

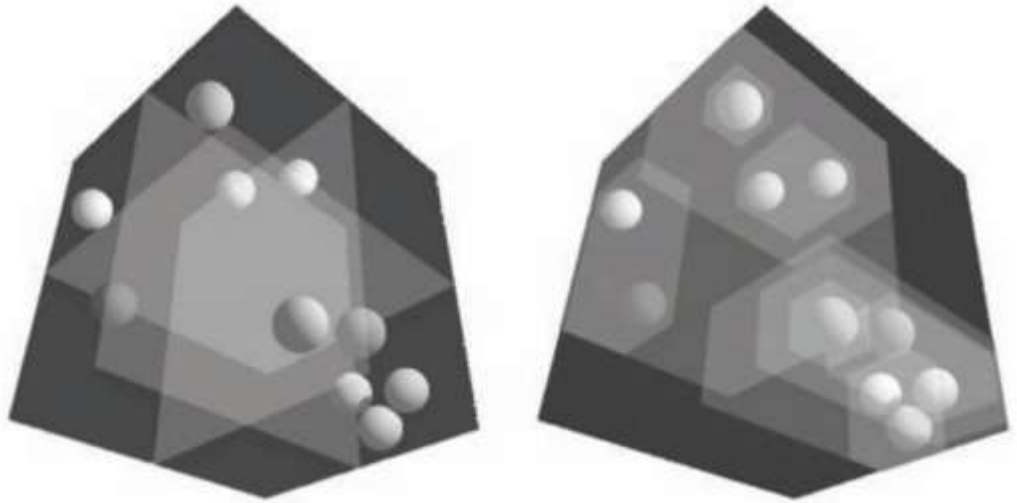
- 그래픽스 연산의 가속화하기 위해 geometric object를 공간상에서 효율적으로 관리하는 자료 구조 필요
- 예시)

- In ray tracing: ray가 object와 충돌하는지 빠르게 테스트
- In computer games, physical simulation: scene에 있는 object들이 서로 충돌하는지 감지
- In culling: viewing frustum안에 있지 않은 object들을 제거하여 화면에 그려지지 않게 함

# Spatial Data Structure의 종류



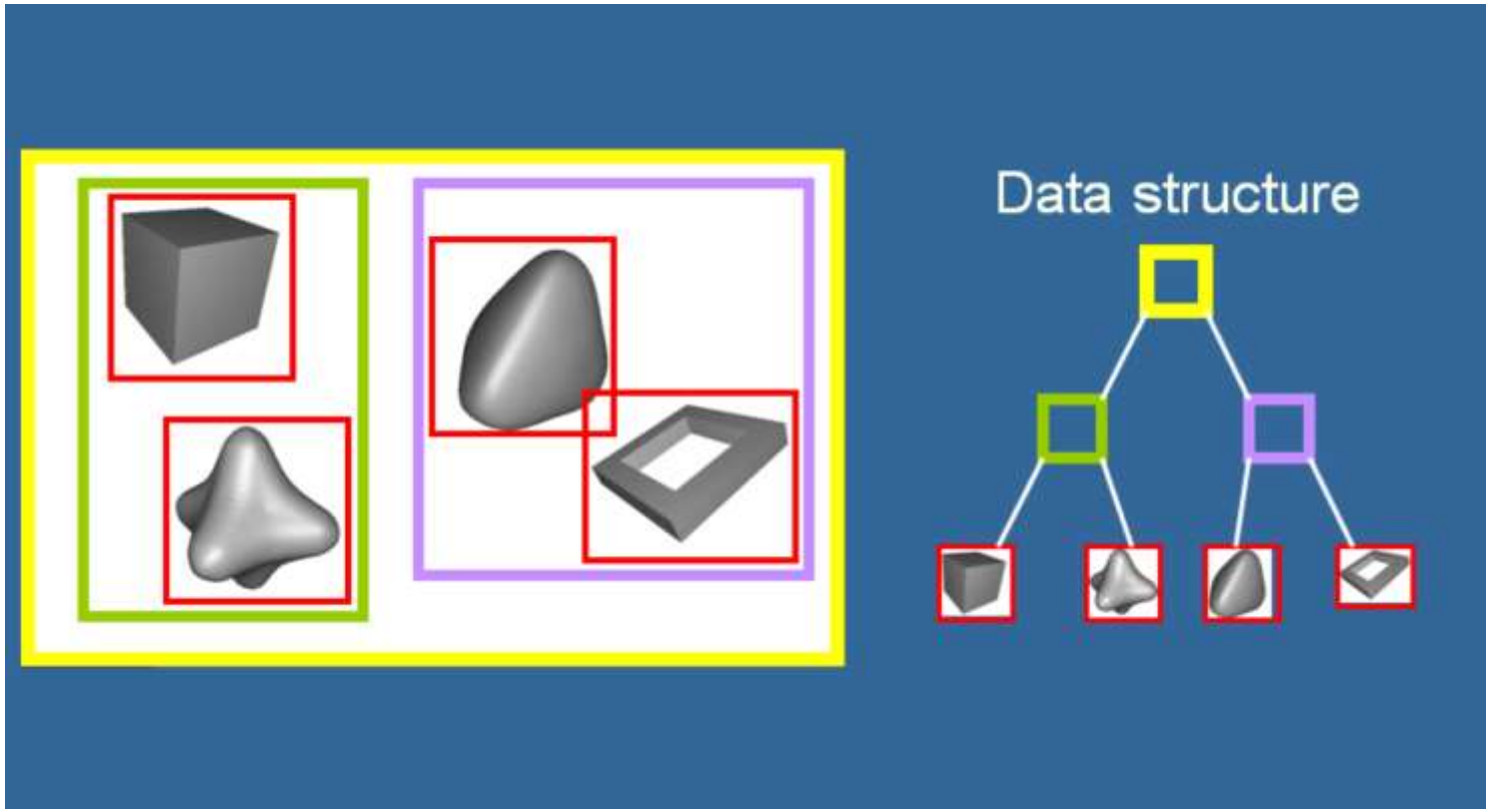
- Object partitioning method
  - 물체를 (겹치지 않는) 그룹으로 나누고, 그룹의 계층(hierarchy)를 구서
  - 각 그룹은 공간상으로는 겹칠 수 있음
- Space partitioning method
  - 공간을 계층적으로 나누고, 나눈 공간에 속하는 물체들을 묶음
  - 한 물체가 여러 partition에 속할 수 있음



(왼쪽) space partitioning의 예 (uniform space partitioning)
(오른쪽) object partitioning의 예 (bounding boxes)

# Object Partitioning: **B**ounding **V**olume **H**ierarchy






- 각 물체를 보다 단순한 물체로 완전히 감쌈
  예) 박스 (axis-aligned box, oriented box), 구, 사면체 등

- Hierarchy의 root부터 tree를 traverse하며 기하 query (i.e. 충돌 감지)를 실행:
  - If answer = No (i.e. no collision): 그 node의 전체 subtree를 skip
  - If answer = Yes (i.e. maybe collision): 그 node의 child node에 대해 재귀적으로 query를 실행

- Bounding volume들을 최대한 tight하게 정의해야 더 많이 가속화됨

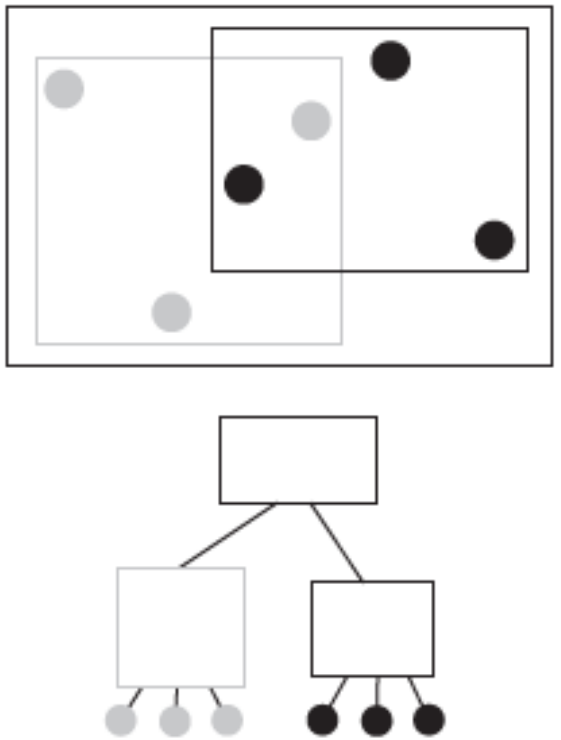
# Axis-Aligned Bounding Box (AABB) 기반의 BVH

# Bounding Volume Hierarchy: (Axis-aligned Bounding Box)



- Hierarchical bounding box – bounding box들의 hierarchy
  - 하나의 bounding box는 그 안에 있는 모든 object들을 항상 bounding
  - Bounding box는 공간 상에서 항상 그 안에 있는 object들을 포함하는 것은 아님
- k-ary tree를 이용해서 표현 (보통 binary tree)

```
BvhNode {
  surface-pointer left;
  surface-pointer right;
  Box bbox;

  virtual box bounding-box()
  virtual bool hit(ray a + tb,
                   real t0, real t1,
                   hit-record rec);
}
```

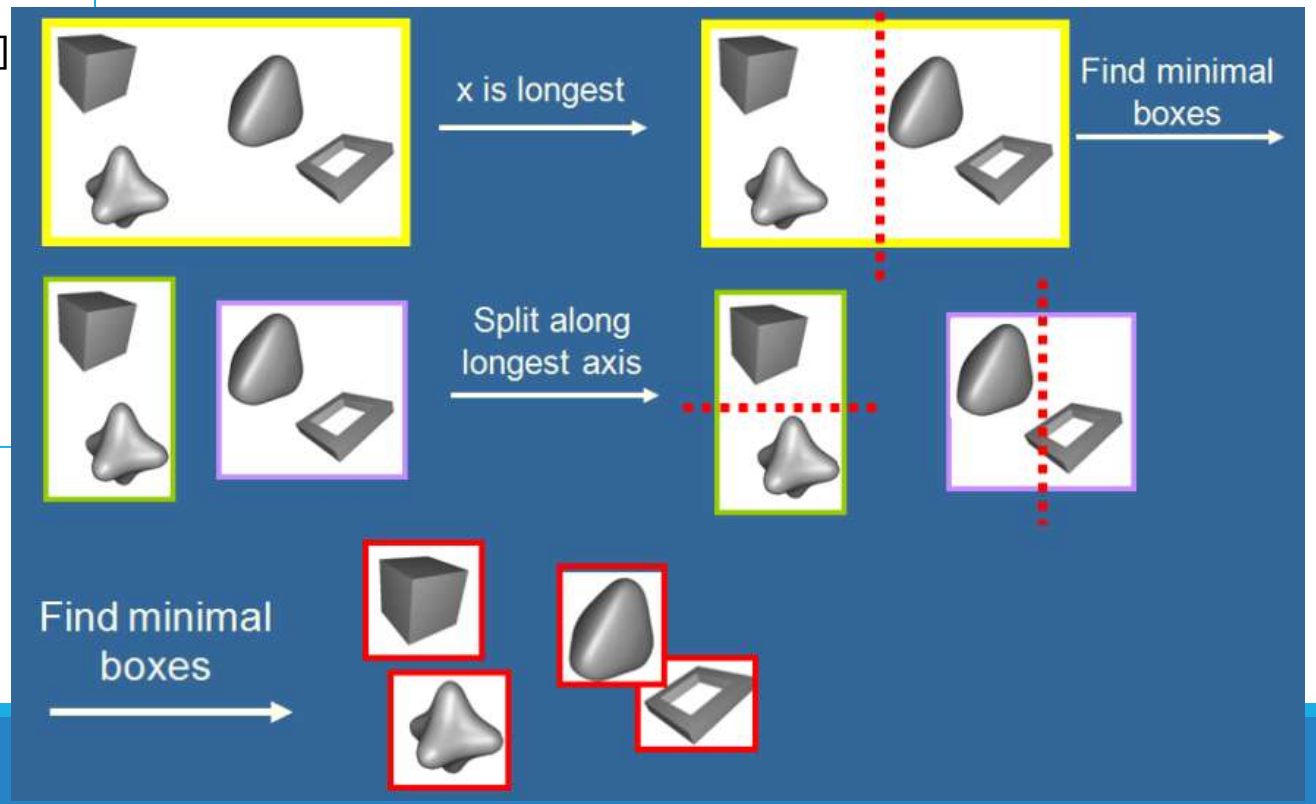


# Bounding Volume Hierarchy: (Axis-aligned Bounding Box)



## Hierarchical Bounding Box Construction

```
Function BvhNode::create(object-array A, int AXIS){
  N = A.length;
  if (N = 1) then
    left = A[0]; right = NULL;
    bbox = bounding-box(A[0])
  else if (N = 2) then
    left = A[0]; right = A[1];
    bbox = combine(bounding-box(A[0]), bounding-box(A[1]
  else
    find the midpoint m of bounding box of A along AXIS
    partition A into lists with length=k and (N-k)
    find the longest axis NEW_AXIS of bounding box of A
    left = new BvhNode(A[0..k], NEW_AXIS);
    right = new BvhNode(A[k+1,..N-1], NEW_AXIS);
    bbox = combine(left→bbox, right →bbox)
}
```



# Let's Implement AABB-based BVH in C++!



```
class AABBBox
{
public:
    vec3 BoxMin;
    vec3 BoxMax;
    int Id;
```

```
class AABBNode
{
public:
    AABBNode *pLeftChild;
    AABBNode *pRightChild;
    const AABBBox **Boxes;
    int NumBoxes;

    vec3 BoxMin;
    vec3 BoxMax;
```

```
class AABBBvh
{
public:
    AABBNode *pRoot;

    /* Leaf geometry information. */
    AABBBox *Boxes;
    AABBBox **BoxPtrs;
        …

protected:
    const TriMesh *pMesh;
        …
```

[BVH 구현에 필요한 자료 구조의 예시]

① AABBBvh: AABB 노드들의 트리를 저장하기 위한 자료 구조
- Boxes: Leaf node (삼각형 1개를 감싸는 박스)들을 저장
- BoxPtrs: Leaf node들은 1차원 배열에 순서대로 저장하여 효율성을 높임
- pRoot: 트리의 root 노드
- pMesh: AABBBvh가 참고하는 원본 삼각 메시

# Let's Implement AABB-based BVH in C++!



```
class AABBBox
{
public:
    vec3 BoxMin;
    vec3 BoxMax;
    int Id;
```

```
class AABBNode
{
public:
    AABBNode *pLeftChild;
    AABBNode *pRightChild;
    const AABBBox **Boxes;
    int NumBoxes;

    vec3 BoxMin;
    vec3 BoxMax;
```

```
class AABBBvh
{
public:
    AABBNode *pRoot;

    /* Leaf geometry information. */
    AABBBox *Boxes;
    AABBBox **BoxPtrs;
        …

protected:
    const TriMesh *pMesh;
        …
```

[BVH 구현에 필요한 자료 구조의 예시]

② AABBNode: 각 AABB를 저장

- pLeftChild, pRightChild: 자식 노드들
- NumBoxes: 현재 노드에 bounding 되는 leaf (삼각형)의 개수
- Boxes: 현재 노드에 해당하는 leaf box들의 포인터
- BoxMin, BoxMax: 현재 노드의 bounding box의 크기

[BVH 구현에 필요한 자료 구조의 예시]

③ AABBBox: Leaf node에 저장하는 box. 하나의 box가 삼각형을 감쌈

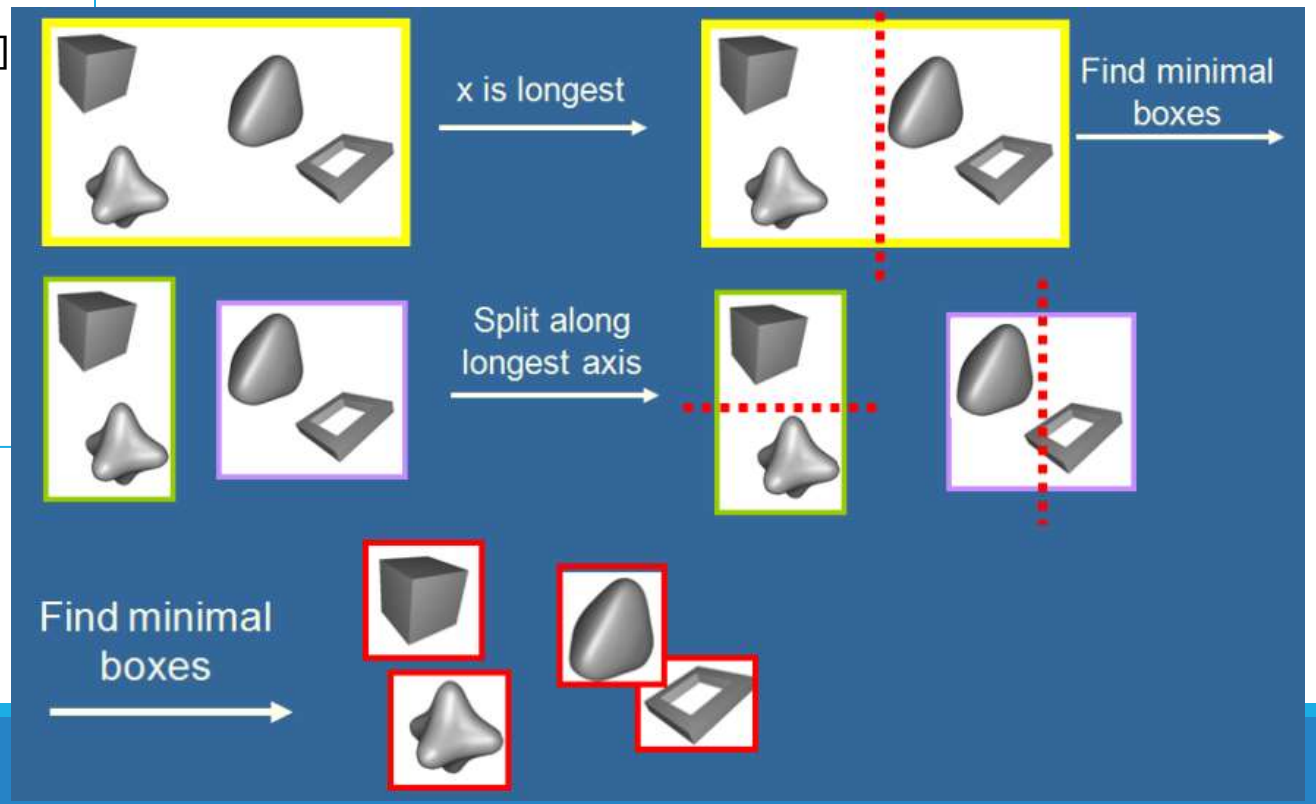
- BoxMin, BoxMax: 해당 leaf node에 속하는 삼각형(들)의 aabb
- Id: leaf box의 인덱스. 원본 메시에서 해당하는 삼각형의 인덱스

# Bounding Volume Hierarchy: (Axis-aligned Bounding Box)



## Hierarchical Bounding Box Construction

```
Function BvhNode::create(object-array A, int AXIS){
  N = A.length;
  if (N = 1) then
    left = A[0]; right = NULL;
    bbox = bounding-box(A[0])
  else if (N = 2) then
    left = A[0]; right = A[1];
    bbox = combine(bounding-box(A[0]), bounding-box(A[1]
  else
    find the midpoint m of bounding box of A along AXIS
    partition A into lists with length=k and (N-k)
    find the longest axis NEW_AXIS of bounding box of A
    left = new BvhNode(A[0..k], NEW_AXIS);
    right = new BvhNode(A[k+1,..N-1], NEW_AXIS);
    bbox = combine(left→bbox, right →bbox)
}
```



# AABB BVH Construction 예시



```
int AABBBvh::BuildBvh(const TriMesh *mesh)
{
    if (mesh == NULL)
        return 0;

    pMesh = mesh;
    int NumTris = pMesh -> NumTris;
    Boxes = new AABBBox[NumTris];
    BoxPtrs = new AABBBox *[NumTris];

    /* Construct aabb for each triangle. */
    for (int i = 0; i < NumTris; i++) {
        Boxes[i].BoxMin = vec3(FLT_MAX, FLT_MAX, FLT_MAX);
        Boxes[i].BoxMax = vec3(-FLT_MAX, -FLT_MAX, -
FLT_MAX);

        for (int j = 0; j < 3; j++) {
            int vIdx = pMesh -> indices[i * 3 + j];
            const vec4 pt = pMesh -> vertices[vIdx];

            for (int Axis = 0; Axis < 3; Axis++) {
                if (pt[Axis] < Boxes[i].BoxMin[Axis])
                    Boxes[i].BoxMin[Axis] = pt[Axis];
                if (pt[Axis] > Boxes[i].BoxMax[Axis])
                    Boxes[i].BoxMax[Axis] = pt[Axis];
            }
        }
```

```
…
/* Make the bounding box slightly larger than the original
triangle. */
        for (int Axis = 0; Axis < 3; Axis++) {
            Boxes[i].BoxMin[Axis] -= EPS;
            Boxes[i].BoxMax[Axis] += EPS;
        }
        Boxes[i].Id = i;
        BoxPtrs[i] = &(Boxes[i]);
    }

    pRoot = new AABBNode;
    pRoot -> SplitNode((const AABBBox **)BoxPtrs, NumTris);

    return 1;
}
```

# AABB BVH Construction 예시

한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING

```
void AABBNode::SplitNode(const AABBBox **boxPtr, int Size)
{
    if (Size == 0)
        return;

    Boxes = boxPtr;
    NumBoxes = Size;

    vec3 Center, Min, Max;

    /* Compute the bounding box of object arrays with Size
triangles. */
    Min = vec3(FLT_MAX, FLT_MAX, FLT_MAX);
    Max = vec3(-FLT_MAX, -FLT_MAX, -FLT_MAX);

    /* Find the midpoint m of bounding box of A along Axis.
*/
    for (int i = 0; i < Size; i++) {
        for (int Axis = 0; Axis < 3; Axis++) {
            if (Boxes[i] -> BoxMin[Axis] < Min[Axis])
                Min[Axis] = Boxes[i] -> BoxMin[Axis];
            if (Boxes[i] -> BoxMax[Axis] > Max[Axis])
                Max[Axis] = Boxes[i] -> BoxMax[Axis];
        }
    }

    BoxMin = Min;
    BoxMax = Max;

    if (Size == 1)
        return;
```

```
    /* When there exists more than one box in this node.*/
    Center = (Min + Max) * 0.5;

    vec3 BoxSize = Max - Center;
    float LBoxSize = BoxSize[0];
    int LAxis = 0;

    for (int Axis = 1; Axis < 3; Axis++) {
        if (LBoxSize < BoxSize[Axis]) {
            LBoxSize = BoxSize[Axis];
            LAxis = Axis;
        }
    }
```

# AABB BVH Construction 예시

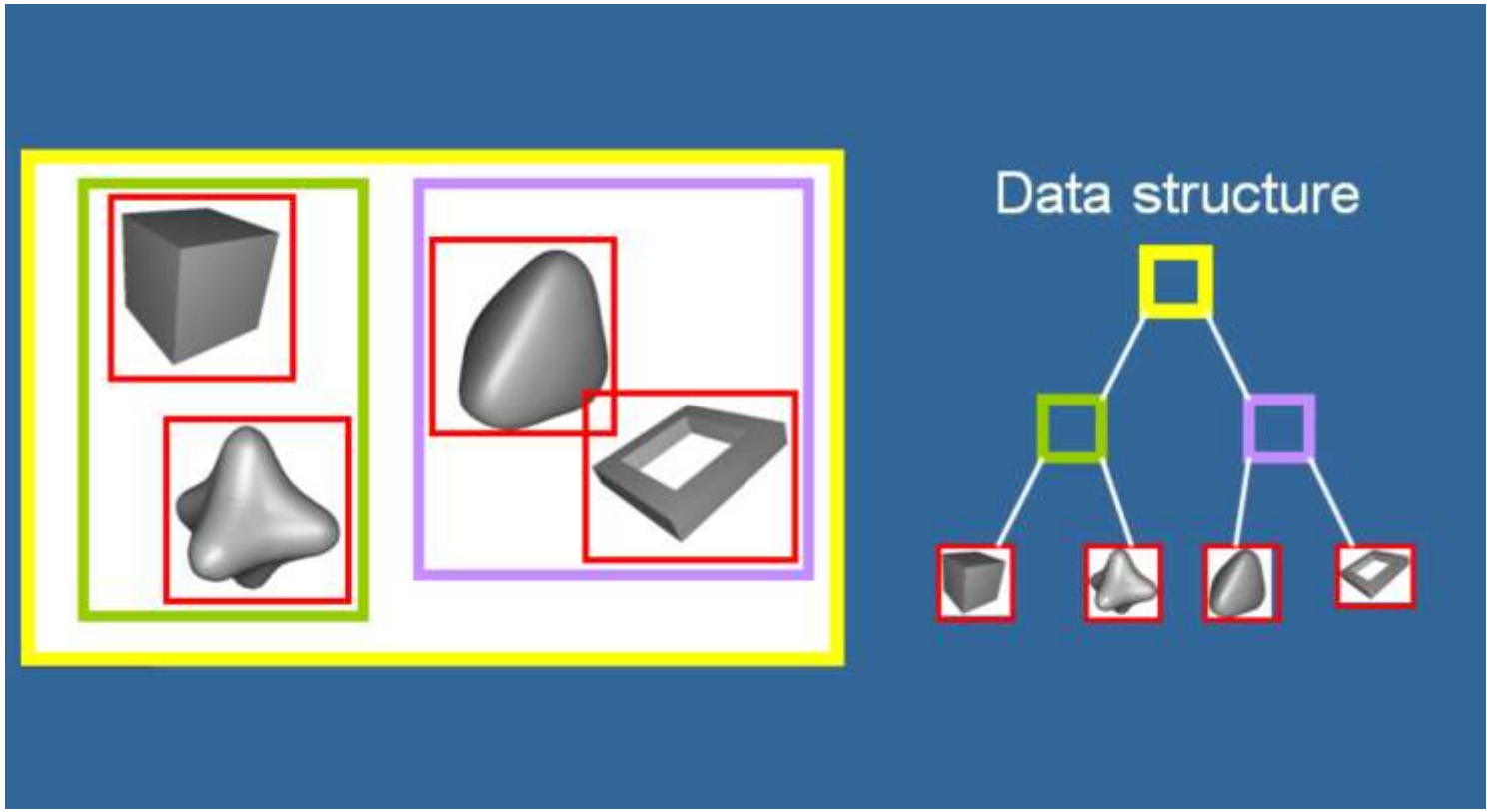
한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING

```
    /* Partition A into lists with length=k and (N-k) */
    float MidVal = Center[LAxis];

    int Lesser = 0,
        Greater = 0,
        Tested = 0,
        k = 0;
    while (Tested < Size) {
        float Val = (Boxes[k]-> BoxMax[LAxis] + Boxes[k]->
BoxMin[LAxis]) * 0.5;
        if (Val > MidVal) {
            /* Move the element to the end of the array. */
            int Idx = Size - 1 - Greater;
            const AABBBox *Temp = Boxes[Idx];
            Boxes[Idx] = Boxes[k];
            Boxes[k] = Temp;
            Greater++;
        }
        else if (Val < MidVal) {
            /* Move the element to the front of the array.
*/
            const AABBBox *Temp = Boxes[Lesser];
            Boxes[Lesser] = Boxes[k];
            Boxes[k] = Temp;
            Lesser++;
            k++;
        }
        else {
            /* Skip to the next element when Val == MidVal
*/
            k++;
        }
        Tested++;
    }
```

```
    /* Distribute the equal element to both lesser and
greater side. */
    int Equal = Size - Lesser - Greater;
    if (Equal % 2 == 0) {
        Lesser += Equal / 2;
        Greater += Equal / 2;
    }
    else {
        if (Lesser > Greater) {
            Lesser += Equal / 2;
            Greater += (Equal / 2 + 1);
        }
        else {
            Lesser += (Equal / 2 + 1);
            Greater += Equal / 2;
        }
    }

    assert(Lesser != 0 && Greater != 0);

    if (Lesser > 0) {
        pLeftChild = new AABBNode;
        pLeftChild -> SplitNode(&(Boxes[0]), Lesser);
    }

    if (Greater > 0) {
        pRightChild = new AABBNode;
        pRightChild -> SplitNode(&(Boxes[Lesser]), Greater);
    }
}
```

# BVH를 이용한 충돌 감지






- 각 물체를 보다 단순한 물체로 완전히 감쌈
  예) 박스 (axis-aligned box, oriented box), 구, 사면체 등

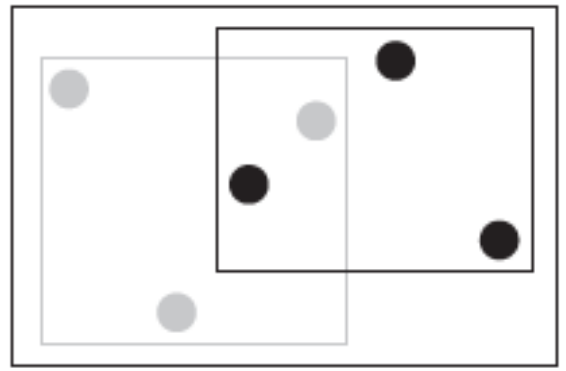
- Hierarchy의 root부터 tree를 traverse하며 기하 query (i.e. 충돌 감지)를 실행:
  - If answer = No (i.e. no collision): 그 node의 전체 subtree를 skip
  - If answer = Yes (i.e. maybe collision): 그 node의 child node에 대해 재귀적으로 query를 실행

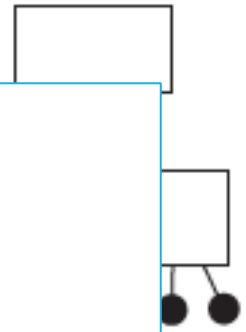
- Bounding volume들을 최대한 tight하게 정의해야 더 많이 가속화됨

# Bounding Volume Hierarchy: (Axis-aligned Bounding Box)



```
Function bool BvhNode::hit(ray a + tb,
                  real t0, real t1, hit-record rec){
  if (bbox.hitbox(a + tb, t0, t1) then
    hit-record lrec, rrec;
    left-hit = (left ≠ NULL) and
                  (left→hit(a + tb, t0, t1, lrec))
    right-hit = (right ≠ NULL) and
                   (right→hit(a + tb, t0, t1, rrec))
    if (left-hit and right-hit) then
      rec = (lrec.t < rrec.t) ? lrec : rrec;
      return true;
    else if (left-hit) rec = lrec; return true;
    else if (right-hit) rec= rrec; return true;
    else false;
  else
    return false;
}
```





```
BvhNode {
  surface-pointer left;
  surface-pointer right;
  Box bbox;

  virtual box bounding-box()
  virtual bool hit(ray a + tb,
           real t0, real t1,
           hit-record rec);
}
```

```
Function bool BvhNode::hit(ray a + tb,
                  real t0, real t1, hit-record rec){
  if (bbox.hitbox(a + tb, t0, t1) then
    hit-record lrec, rrec;
    left-hit = (left ≠ NULL) and
                  (left→hit(a + tb, t0, t1, lrec))
    right-hit = (right ≠ NULL) and
                   (right→hit(a + tb, t0, t1, rrec))
    if (left-hit and right-hit) then
      rec = combine(lrec, rrec)
      return true;
    else if (left-hit) rec = lrec; return true;
    else if (right-hit) rec= rrec; return true;
    else false;
  else
    return false;
}
```

# Ray-Box Intersection



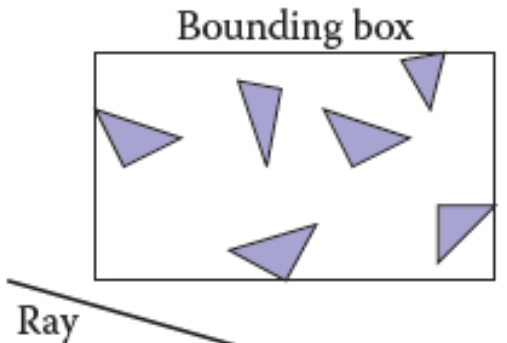
- 2D Bounding box – 다음의 4 선분으로 둘러싸인 2D 영역
  - $x = x_{min},\ x = x_{max},\ y = y_{min},\ y = y_{max}$
- Q) 주어진 ray와 a bounding box가 충돌하는가?

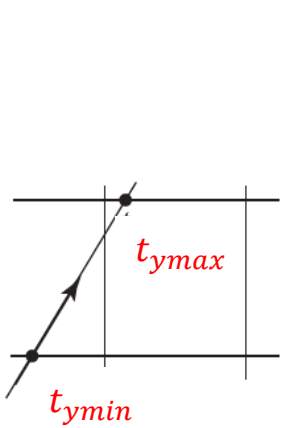
⇒ intersection intervals 검사

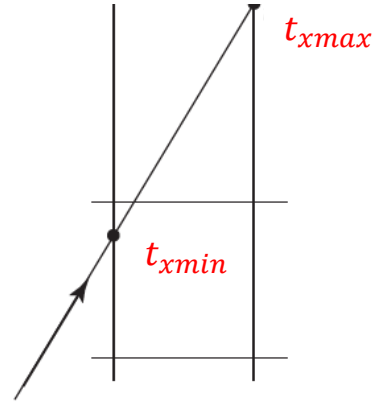
- Ray 상의 한 점의 식: $p(t) = (x_e, y_e) + (x_d, y_d)t$
- Finding t's that intersect $x = x_{min},\ x = x_{max}$

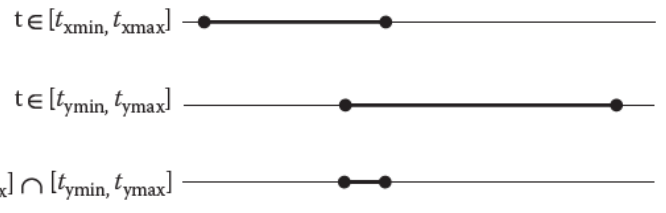
  $t_{xmin} = \frac{x_{min} - x_e}{x_d},\ t_{xmax} = \frac{x_{max} - x_e}{x_d}$

  => 만약 $t \in [t_{xmin}, t_{xmax}]$, 이 점은 두 수직선 사이에 위치
- 같은 식으로 만약 $t \in [t_{ymin}, t_{ymax}]$, 이 점은 두 수평선 사이에 위치
- $t \in [t_{xmin}, t_{xmax}], t \in [t_{ymin}, t_{ymax}]$를 동시에 만족할 때 이 점은 2D bounding box 안에 위치



# Ray-Object Intersection: Ray-Sphere



- 구의 방정식:

$$f(\boldsymbol{x}) = (x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - R^2 = (\boldsymbol{x} - \boldsymbol{c}) \cdot (\boldsymbol{x} - \boldsymbol{c}) - R^2 = 0$$

- Ray $\boldsymbol{p}(t) = \boldsymbol{e} + t\boldsymbol{d}$와 구가 만나려면,

$f(\boldsymbol{p}(t)) = 0$

$\Rightarrow f(\boldsymbol{e} + t\boldsymbol{d}) = 0$

$\Rightarrow (\boldsymbol{e} + t\boldsymbol{d} - \boldsymbol{c}) \cdot (\boldsymbol{e} + t\boldsymbol{d} - \boldsymbol{c}) - R^2 = 0$

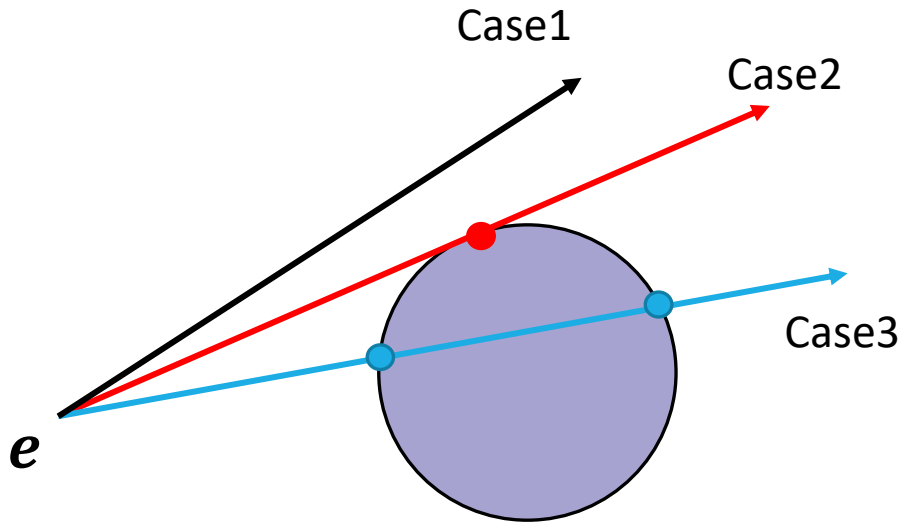
$\Rightarrow \underbrace{(\boldsymbol{d} \cdot \boldsymbol{d})}_{A} t^2 + \underbrace{2\boldsymbol{d} \cdot (\boldsymbol{e} - \boldsymbol{c})}_{B} t + \underbrace{(\boldsymbol{e} - \boldsymbol{c}) \cdot (\boldsymbol{e} - \boldsymbol{c}) - R^2}_{C} = 0$

Case 1. $B^2 - 4AC < 0$: 식의 해 없음 => Ray가 구와 만나지 않음

Case 2. $B^2 - 4AC = 0$ : Ray와 구가 한 점에서 만남 ($t = -\frac{B}{2A}$)

Case 3. $B^2 - 4AC > 0$ : Ray와 구가 두 점에서 만남($t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$)

- Ray가 구와 만날 때의 normal $\boldsymbol{n} = \frac{\boldsymbol{p} - \boldsymbol{c}}{R}$



# Ray-Object Intersection: Ray-Triangle



- 세 점 $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$로 이루어진 삼각형의 내부의 점 $\boldsymbol{f}(u, v)$:

$$\boldsymbol{f}(u, v) = \boldsymbol{a} + \beta(\boldsymbol{b} - \boldsymbol{a}) + \gamma(\boldsymbol{c} - \boldsymbol{a}), \qquad \beta > 0, \gamma > 0, \beta + \gamma < 1$$

- Ray $\boldsymbol{p}(t) = \boldsymbol{e} + t\boldsymbol{d}$가 삼각형의 내부를 지나기 위해서는

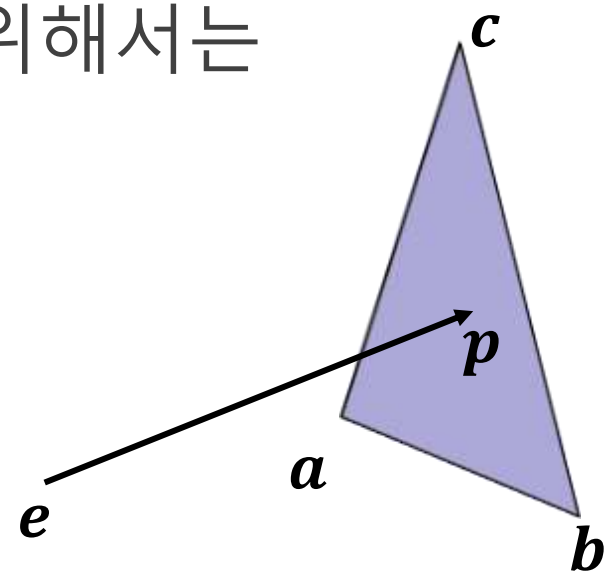
$$\boldsymbol{e} + t\boldsymbol{d} = \boldsymbol{f}(u, v)$$

$$\Rightarrow x_e + t x_d = x_a + \beta(x_b - x_a) + \gamma(x_c - x_a)$$

$$y_e + t y_d = y_a + \beta(y_b - y_a) + \gamma(y_c - y_a)$$

$$z_e + t z_d = z_a + \beta(z_b - z_a) + \gamma(z_c - z_a)$$

의 선형 시스템의 해 $(\beta, \gamma)$가 위의 조건을 만족



# Ray-Object Intersection: Ray-Polygon



- Polygon: $\boldsymbol{p}_1, \ldots \boldsymbol{p}_m$의 점들로 구성, 한 평면 위에 있음 (법선 $\boldsymbol{n}$)
- Step 1: Ray $\boldsymbol{p}(t) = \boldsymbol{e} + t\boldsymbol{d}$ 가 polygon을 포함한 평면과 만날 때의 $t = t_p$ 계산

$$\left(\boldsymbol{e} + t_p\boldsymbol{d} - \boldsymbol{p}_1\right) \cdot \boldsymbol{n} = 0 \rightarrow t_p = \frac{(\boldsymbol{p}_1 - \boldsymbol{e}) \cdot \boldsymbol{n}}{\boldsymbol{d} \cdot \boldsymbol{n}}$$

- Step 2: $\boldsymbol{p}(t_p)$가 polygon의 내부에 있는지 test
  - Ex) 점 $\boldsymbol{p}(t_p)$과 polygon을 xy-plane으로 투영하고 점에서 2D ray를 그림. 이 ray가 polygon과 홀수 번 만나면 점은 내부에 있고, 짝수 번 만나면 점은 외부에 있음

# Programming Assignment #4

# Assignment4: Bounding Volume Hierarchy

한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING

- 문제 개요: AABB를 기반으로 하는 BVH(Bounding volume hierarchy)를 만들어서 입력 삼각 메시를 감싸고, BVH를 이용해서 기하 연산을 가속화하기

1. 삼각 메시 (예: bunny_normals.obj)가 주어질 때 이를 감싸는 AABB 기반의 BVH를 계산하고, 화면에 표시하기

2. BVH를 이용해서 다음의 기하 연산들을 수행하기
   ① 기하 연산 1. Ray-mesh intersection: 임의의 Ray들이 삼각 메시와 충돌할 때 충돌하는 지점의 삼각형과, 삼각형 위에서 충돌하는 점을 찾고, 이를 화면에 표시하기
   ② 기하 연산 2. Plane-mesh intersection: 임의의 Plane이 삼각 메시와 충돌할 때 충돌하는 지점의 삼각형과 그 삼각형을 지나는 충돌 선을 찾고, 이를 화면에 표시하기

# Assignment4: Bounding Volume Hierarchy

한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING

1. AABB기반의 BVH를 계산하고 화면에 표시하기
   - ‘s’ 버튼을 누르면 AABB가 삼각 메시와 함께 그려지고 다시 ‘s’를 누르면 삼각 메시만 그림
   - BVH가 보이는 상태에서는 위 화살표 키를 누르면 깊은 depth를 가진 AABB Node들이 그려지고, 아래 화살표 키를 누르면 그리는 AABB Node의 depth를 감소시킴

# Assignment4: Bounding Volume Hierarchy

한양대학교 ERICA
소프트웨어융합대학
COLLEGE OF COMPUTING

## 2. BVH를 이용하여 충돌 연산 가속화하기

### ① Ray-Mesh intersection

- 1을 눌러, 입력으로 주어질 ray의 개수=n 을 사용자 입력으로 받음
- n개의 광선에 대하여 각각의 광선이 삼각 메시와 충돌할 때, 충돌하는 지점의 삼각형들과 삼각형 위의 충돌점들의 좌표를 계산함
- 충돌 삼각형과 충돌 점을 메시 위에 다른 색상으로 표시함

- 사용자가 x, y, z키를 누르면 광선의 시작점을 x축, y축, z축으로 조금씩 평행 이동시키고, 이동된 점에서 다시 임의의 방향으로 n개의 광선을 생성, 충돌 정보를 갱신함

# Assignment4: Bounding Volume Hierarchy



## 2. BVH를 이용하여 충돌 연산 가속화하기

### ② Plane-Mesh intersection

- 2를 누르면 임의의 법선을 가진 임의의 평면을 random으로 생성 (평면이 메시 내부의 한 점을 지나도록 생성하여 Plane-Mesh 충돌을 관찰할 수 있도록 한다.)
- 생성된 평면과 삼각 메시가 충돌할 때, 충돌하는 지점의 삼각형들과 삼각형 위의 충돌 선들을 계산한다.
- 충돌 삼각형과 충돌 선들을 메시 위에 다른 색상으로 표시함

- 사용자가 x, y, z키를 눌러서 평면을 x축, y축, z축으로 이동시키고, 이동된 점에서 다시 임의의 법선 방향을 가진 새 평면을 생성하여 충돌 정보를 갱신함

# Assignment4: Bounding Volume Hierarchy



- 참고 사항
  - 충돌 연산은 AABB를 사용하지 않아도 구현할 수 있다. 하지만 충돌 연산을 포함한 과제의 실행 프로그램이 채점하는 컴퓨터에서 적당한 속도로 실행되어야 함
  - 충돌하는 삼각형을 먼저 찾아서 화면에 표시하고, 그 다음에 충돌 점(충돌 선)을 계산하는 것을 추천

# Assignment 4



- 제출물
  - 프로그램 컴파일에 필요한 모든 h 파일, cpp 파일 및 shader 파일(glsl). 단 수업 중에 배포한 h 파일들은 제출하지 않아도 됨
  - 실행 파일(exe) 및 실행 파일의 화면 캡쳐 파일들
  - Readme.txt 구현 상의 특이점, 실행 프로그램 사용법 등을 서술

- 마감일: 12월 15일 금, 23시 59분