Data Structures & Algorithms for Geometry

- Agenda:
  - Introduce course
  - Linear algebra primer
  - Introduce bounding volumes
    - General BV characteristics
    - Axis-aligned bounding boxes

What should you already know?

- C++ and object oriented programming

What should you already know?

- C++ and object oriented programming
  - For most assignments you will need to implement classes that conform to a very specific interface.

What should you already know?

- C++ and object oriented programming
  - For most assignments you will need to implement classes that conform to a very specific interface.
- Fundamental data structures

What should you already know?

- C++ and object oriented programming
  - For most assignments you will need to implement classes that conform to a very specific interface.
- Fundamental data structures
  - Most data structures for geometry are specialized versions of linked lists, binary trees, etc.

What should you already know?

- C++ and object oriented programming
  - For most assignments you will need to implement classes that conform to a very specific interface.
- Fundamental data structures
  - Most data structures for geometry are specialized versions of linked lists, binary trees, etc.
- Some knowledge of linear algebra / vector math.

What should you already know?

- C++ and object oriented programming
  - For most assignments you will need to implement classes that conform to a very specific interface.
- Fundamental data structures
  - Most data structures for geometry are specialized versions of linked lists, binary trees, etc.
- Some knowledge of linear algebra / vector math.
  - Can probably pick most of it up on the way, but be prepared to work a little harder.

What will you learn?

- Creation and operations on bounding volumes.

What will you learn?

- Creation and operations on bounding volumes.
  - Bounding sphere, AABB, and OBB may be gibberish now, but they'll be second nature soon.

What will you learn?

- Creation and operations on bounding volumes.
  - Bounding sphere, AABB, and OBB may be gibberish now, but they'll be second nature soon.
- Creation, traversal, and operations on bounding volume hierarchies.

What will you learn?

- Creation and operations on bounding volumes.
  - Bounding sphere, AABB, and OBB may be gibberish now, but they'll be second nature soon.
- Creation, traversal, and operations on bounding volume hierarchies.
  - This is where familiarity with tree structures is important.

What will you learn?

- Creation and operations on bounding volumes.
  - Bounding sphere, AABB, and OBB may be gibberish now, but they'll be second nature soon.
- Creation, traversal, and operations on bounding volume hierarchies.
  - This is where familiarity with tree structures is important.
- Creation and traversal of space partitions.

What will you learn?

- Creation and operations on bounding volumes.
  - Bounding sphere, AABB, and OBB may be gibberish now, but they'll be second nature soon.
- Creation, traversal, and operations on bounding volume hierarchies.
  - This is where familiarity with tree structures is important.
- Creation and traversal of space partitions.
  - Doom (the *original*) made BSP trees popular...now you get to implement one.

What will you learn? (cont.)

- Representation and storage of polygons.

What will you learn? (cont.)

- Representation and storage of polygons.
  - May seem trivial, but what if you want to operate on all the polygons that share a vertex?

What will you learn? (cont.)

- Representation and storage of polygons.
  - May seem trivial, but what if you want to operate on all the polygons that share a vertex?
- Numerical stability and geometrical robustness.

What will you learn? (cont.)

- Representation and storage of polygons.
  - May seem trivial, but what if you want to operate on all the polygons that share a vertex?
- Numerical stability and geometrical robustness.
  - Floating point math is inexact. This can cause problems. *Big* problems.

How will you be graded?

- Bi-weekly quizzes worth 5 points each.
- A final exam worth 50 points.
- Bi-weekly programming assignments with 10 points each.
- A term project worth 50 points.

How will programs be graded?

- First and foremost, does the program produce the correct output?
- Are appropriate algorithms and data-structures used?
- Is the code readable and clear?

Linear algebra primer

- Three important data types:
  - Scalar values
  - Row / column vectors
    - 1x4 and 4x1 are the sizes we'll most often encounter
  - Square matrices
    - 4x4 is the size we'll most often encounter

Scalars

- These are the numbers you know!
  - Example: 3.14, 5.0, 99.9, $\sqrt{2}$, etc.

Row vectors

- These are special matrices that have multiple columns but only one row.
  - Example: $\begin{bmatrix} 5.0 & 3.14 & 37 \end{bmatrix}$
- Add and subtract the way you would expect.
  - Example: $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} + \begin{bmatrix} 9 & 10 & 11 \end{bmatrix} = \begin{bmatrix} 10 & 12 & 14 \end{bmatrix}$
  - Both vectors *must* be the same size.
- Operate with scalars the way you would expect.
  - Example: $3.2 \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 3.2 & 6.4 & 9.6 \end{bmatrix}$
- Notice that vector multiplication is missing...

Column vectors

- These are special matrices that have multiple rows but only one column.
  - Example: $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$
- Work just like row vectors.

Vector operations

- There are only a few operations specific to vectors that are *really* important for us.

Dot Product

- Noted as a "dot" between two vectors (e.g., $A \cdot B$)
- Also known as "inner product."
- Multiply matching elements, sum all the results.
  - Example:

$$\begin{bmatrix} 2.3 & 1.2 \end{bmatrix} \cdot \begin{bmatrix} 1.7 & 6.5 \end{bmatrix} = (2.3 * 1.7) + (1.2 * 6.5) = 11.71$$

Why is the dot product so interesting?

- In 3-space, the dot of two unit vectors is the cosine of the angle between the two vectors.
  - If the vectors are not already normalized (unit length), we can divide the dot product by the magnitudes.
  - Example:

$$\frac{a \cdot b}{|a||b|} = \cos\theta$$

Vector Magnitude

- Noted by vertical bars, like absolute value.
- Take the square root of the dot product of the vector with itself...like absolute value.
- Result is the magnitude (a.k.a. length) of the vector.
  - Example: $$\left\| \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \right\| = \sqrt{\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \cdot \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}} =$$
$$\sqrt{\left(\frac{\sqrt{2}}{2}\right)^2 + \left(\frac{\sqrt{2}}{2}\right)^2} = \sqrt{\frac{2}{4} + \frac{2}{4}} = 1$$

Normalize

- Noted by dividing a vector by its magnitude.
  - Example: $\dfrac{A}{|A|}$
- Results in a vector with the same direction, but a magnitude of 1.0.
- Works the same as with scalars.

Cross Product

- Noted as an X between two vectors (e.g., $a \times b$)
- Derivation of the cross product is not important. The math is:

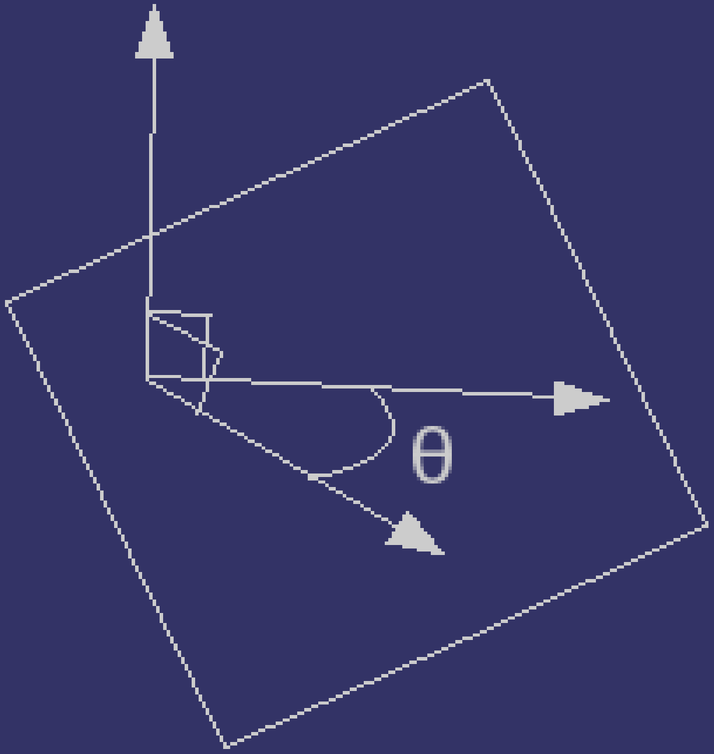
$$a \times b = \left[a_y b_z - a_z b_y \quad a_z b_x - a_x b_z \quad a_x b_y - a_y b_x \right]$$

- *Only* valid in 3-dimensions.

Why is the cross product so interesting?

- Two really useful properties.
  - The result of the cross product between two vectors is a new vector that is perpendicular (also called normal) to both vectors.
  - If the source vectors are normalized:

$$|a \times b| = \sin\theta$$



Matrices

- Like vectors, but have multiple rows and columns.
  - Example: $$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$
- Add and subtract like you would expect.
  - Like vectors, both matrices must be the same size...in both dimensions.

Matrix / vector multiplication

- Special rules apply that make it different from scalar multiplication.
  - **Not** commutative! e.g., $M \times N \neq N \times M$
  - Is associative. e.g., $(NM)P = N(MP)$
  - Column count of first matrix must match row count of second matrix.
    - If M is a 4-by-3 matrix and N is a 3-by-1 matrix, we can do $M \times N$, but *not* $N \times M$.
  - If the source matrices are n-by-m and m-by-p, the resulting matrix will be n-by-p.

Matrix / vector multiplication (cont.)

- To calculate an element of the matrix, C, resulting from AB:

$$C_{ij}=\Sigma_{r=1}^{n} a_{ir} b_{rj}$$

- What does this look like?

Matrix / vector multiplication (cont.)

- To calculate an element of the matrix, C, resulting from AB:

$$C_{ij} = \Sigma_{r=1}^{n} a_{ir} b_{rj}$$

- What does this look like?
  - The dot product of a row of A with a column of B.
  - This is why the column count of A must match the row count of B...otherwise the dot product wouldn't work.

Transpose

- Noted by a "T" in the exponent position (e.g., M^T).
- The rows become the columns, and the columns become the rows.
  - Example:

$$\begin{bmatrix} 2 & 3 \\ 4 & 5 \\ 6 & 7 \end{bmatrix}^T = \begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \end{bmatrix}$$

References

http://en.wikipedia.org/wiki/Matrix_multiplication

http://en.wikipedia.org/wiki/Dot_product

http://en.wikipedia.org/wiki/Cross_product

Break

Bounding Volumes

- From Wikipedia:

  “...a bounding volume for a set of objects is a closed volume that completely contains the union of the objects in the set.”

- Why is this useful?

Bounding Volumes

- From Wikipedia:

  "...a bounding volume for a set of objects is a closed volume that completely contains the union of the objects in the set."

- Why is this useful?
  - We can represent complex geometry (a character model with 50,000 polygons) with a simplified approximation (a box with 6 polygons) that can be tested more quickly.
  - Since the representation is inexact, so is the test result.

Desirable BV Characteristics

- Inexpensive intersection tests

Desirable BV Characteristics

- Inexpensive intersection tests
  - The whole point of using a BV instead of the source geometry is to speed up rejection / acceptance tests between geometric objects.

Desirable BV Characteristics

- Inexpensive intersection tests
- Tight fitting to source geometry

Desirable BV Characteristics

- Inexpensive intersection tests
- Tight fitting to source geometry
  - If the BV is a poor representation of the source geometry, tests between BVs will result in many false positives or false negatives.

Desirable BV Characteristics

- Inexpensive intersection tests
- Tight fitting to source geometry
- Inexpensive to compute

Desirable BV Characteristics

- Inexpensive intersection tests
- Tight fitting to source geometry
- Inexpensive to compute
  - If the BV is too expensive to compute, it may cancel out any speed-up that it provides.

Desirable BV Characteristics

- Inexpensive intersection tests
- Tight fitting to source geometry
- Inexpensive to compute
- Easy to transform

Desirable BV Characteristics

- Inexpensive intersection tests
- Tight fitting to source geometry
- Inexpensive to compute
- Easy to transform
  - Most interesting objects move. If the object moves, its BV needs to move with it. If it is too difficult / expensive to move, it may cancel out the speed-up.

Desirable BV Characteristics

- Inexpensive intersection tests
- Tight fitting to source geometry
- Inexpensive to compute
- Easy to transform
- Inexpensive to store

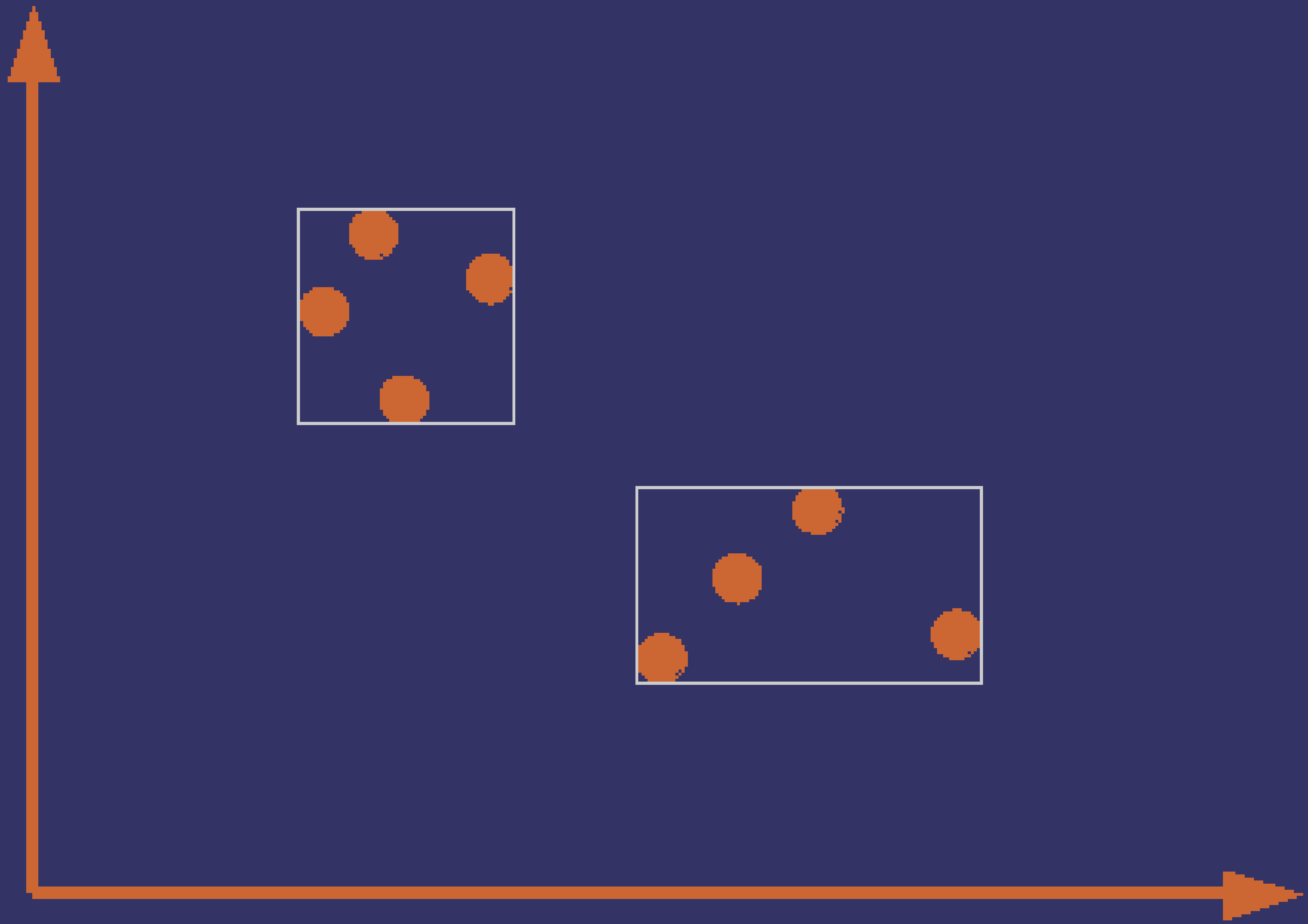
Desirable BV Characteristics

- Inexpensive intersection tests
- Tight fitting to source geometry
- Inexpensive to compute
- Easy to transform
- Inexpensive to store
  - If the BV requires too much space to store or too much time to access, it can negatively impact performance.

Axis-aligned bounding box

- One common BV is the *axis-aligned bounding box* (AABB).
- Just an n-dimensional box whose sides are parallel to the axis and encloses all points.

AABB Example



AABB representation

- Three common ways to represent an AABB
  - All are equivalent, and each can easily be converted to the other.
  - Depending on the data used to create the AABB, one representation may be easier to create than another.

Minimum / maximum point

```
class aabb_m_m {
    // Point such that for every
    // point P in the object:
    //   (min.x <= P.x <= max.x) &&
    //   (min.y <= P.y <= max.y) &&
    //   (min.z <= P.z <= max.z)
    point min;
    point max;
};
```

Min point / diameter

```
class aabb_m_d {
    // Point such that for every
    // point P in the object:
    //   (min.x <= P.x) &&
    //   (min.y <= P.y) &&
    //   (min.z <= P.z)
    point min;

    // Size AABB in each dimension.
    point diameter;
};
```

Center / radius

```
class aabb_c_r {
    // Center of the AABB.
    point center;

    // Distance from 'center' to each
    // side of the AABB.
    point radius;
};
```

AABB-AABB Intersection

- AABB-AABB intersection is just an interval overlap test extended to *n*-dimensions.
  - If there is *no* overlap in *any* dimension, the AABBs cannot intersect.
- Examples:
  - Do A = { c = { 1, 1 }, r = { 1, 2 } } and B = { c = { 3, 4 }, r = { 1, 1 } } intersect?

AABB-AABB Intersection

- AABB-AABB intersection is just an interval overlap test extended to *n*-dimensions.
  - If there is *no* overlap in *any* dimension, the AABBs cannot intersect.
- Examples:
  - Do A = { c = { 1, 1 }, r = { 1, 2 } } and B = { c = { 3, 4 }, r = { 1, 1 } } intersect?
  - Do A = { c = { 1, 1 }, r = { 2, 2 } } and B = { c = { 3, 4 }, r = { 2, 2 } } intersect?

AABB-AABB Intersection (cont.)

```
bool aabb_c_r::intersect(aabb_c_r &box)
{
    const point dist = center - box.center;
    const point rad = radius + box.radius;

    if (abs(dist[0]) > (rad[0])) return false;
    if (abs(dist[1]) > (rad[1])) return false;
    if (abs(dist[2]) > (rad[2])) return false;

    return true;
}
```

AABB Creation

- Creating an initial AABB from source data is a trivial O(n) problem.

AABB Creation

- Creating an initial AABB from source data is a trivial O(n) problem.
  - Scan all of the points tracking the minimum and maximum value in each dimension. Convert the resulting values to the desired representation.
- As the object moves, how can we update the AABB?
  - Rotations are the problem.

AABB from Bounding Sphere

- For a given rotation:
  1. Create a sphere centered at the point of rotation that encompasses the object.
  2. Create an AABB from that sphere.
- Merits and drawbacks of this technique?

AABB from Bounding Sphere

- For a given rotation:
  1. Create a sphere centered at the point of rotation that encompasses the object.
  2. Create an AABB from that sphere.
- Merits and drawbacks of this technique?
  - Fast to compute.
  - Only works well if the object has a single pivot point.
  - Creates a *very* loose AABB.
  - Why not just use the bounding sphere?!?

AABB from Source Data

- Recalculate the AABB from the transformed source data.
  - Don't actually transform the points. Instead transform the axis use for the comparisons.
- Merits and drawbacks of this technique?

AABB from Source Data

- Recalculate the AABB from the transformed source data.
  - Don't actually transform the points. Instead transform the axis use for the comparisons.
- Merits and drawbacks of this technique?
  - Creates a tight fitting AABB.
  - O(n) per transformation can be too expensive.
    - Can be optimized using other search structures and / or a convex hull.
    - More about convex hulls next week.

AABB by Hill-climbing

- Track the six points at the extrema of the AABB
- To update, examine the neighboring points to search for the new extrema.
- Merits and drawbacks of this technique?

AABB by Hill-climbing

- Track the six points at the extrema of the AABB
- To update, examine the neighboring points to search for the new extrema.
- Merits and drawbacks of this technique?
  - Creates a tight fitting AABB.
  - Fast, but...
  - Requires precalculation of a convex hull
  - Requires a data structure that stores connectivity among points

AABB from Rotated AABB

- Transform the original AABB and compute its AABB.
- Merits and drawbacks of this technique?

AABB from Rotated AABB

- Transform the original AABB and compute its AABB.
- Merits and drawbacks of this technique?
  - Fast.
  - Not a very tight fitting AABB.
  - *Very* commonly used.

References

http://en.wikipedia.org/wiki/Bounding_volume

Next week...

- More bounding volumes...
  - Bounding spheres
  - Oriented bounding boxes
  - k-DOPs
  - Convex hulls
- Something we can do with our BVs!
- First programming assignment will be assigned.

Legal Statement

- This work represents the view of the authors and does not necessarily represent the view of IBM or the Art Institute of Portland.
- OpenGL is a trademark of Silicon Graphics, Inc. in the United States, other countries, or both.
- Khronos and OpenGL ES are trademarks of the Khronos Group.
- Other company, product, and service names may be trademarks or service marks of others.