DEPARTMENT OF INFORMATION SYSTEMS AND COMPUTER SCIENCE

Polygons

Convex Shapes and
the Separating Axis Theorem

Lecture Time!

- Polygons: In SFML
- Separating Axis Theorem: Convex Shapes Only

Polygons

- A convex shape in SFML defined by a collection of points in local coordinate space
  - Two consecutive points, as well as the last and first points, form an edge
  - Can define points in either clockwise or counterclockwise order
    - But stick to one order for all polygons in your code/data
- Also has position and rotation

Polygons

```
sf::ConvexShape poly;
// set number of points
// (ex. 3 for a triangle)
poly.setPointCount( NUM_POINTS );
for( int i = 0; i < NUM_POINTS; i++ )
{
    // use vectors to set points
    // in local space
    sf::Vector2f vect;
    poly.setPoint( i, vect );
}
```

DISCS

Polygons

```
// assuming you want a hollow shape
poly.setFillColor(
    sf::Color( 0, 0, 0, 0 ) );
poly.setOutlineColor(
    sf::Color( 255, 255, 255 ) );
poly.setOutlineThickness( -1 );

// drawing is still the same
window.draw( poly );
```

Polygons

- A *convex* polygon has interior angles that are all less than 180 degrees
- A *concave* polygon is not convex
  - Has at least one interior angle that is more than 180 degrees

Separating Axis Theorem

- Project both polygons on the normal of each edge of both polygons
  - If the projections do not overlap for at least one edge, then there is no overlap
  - Otherwise, the polygons overlap
- Requires several operations even if it has early exit possibilities
  - Use AABB or any other "cheap" overlap test first before using SAT

DISCS

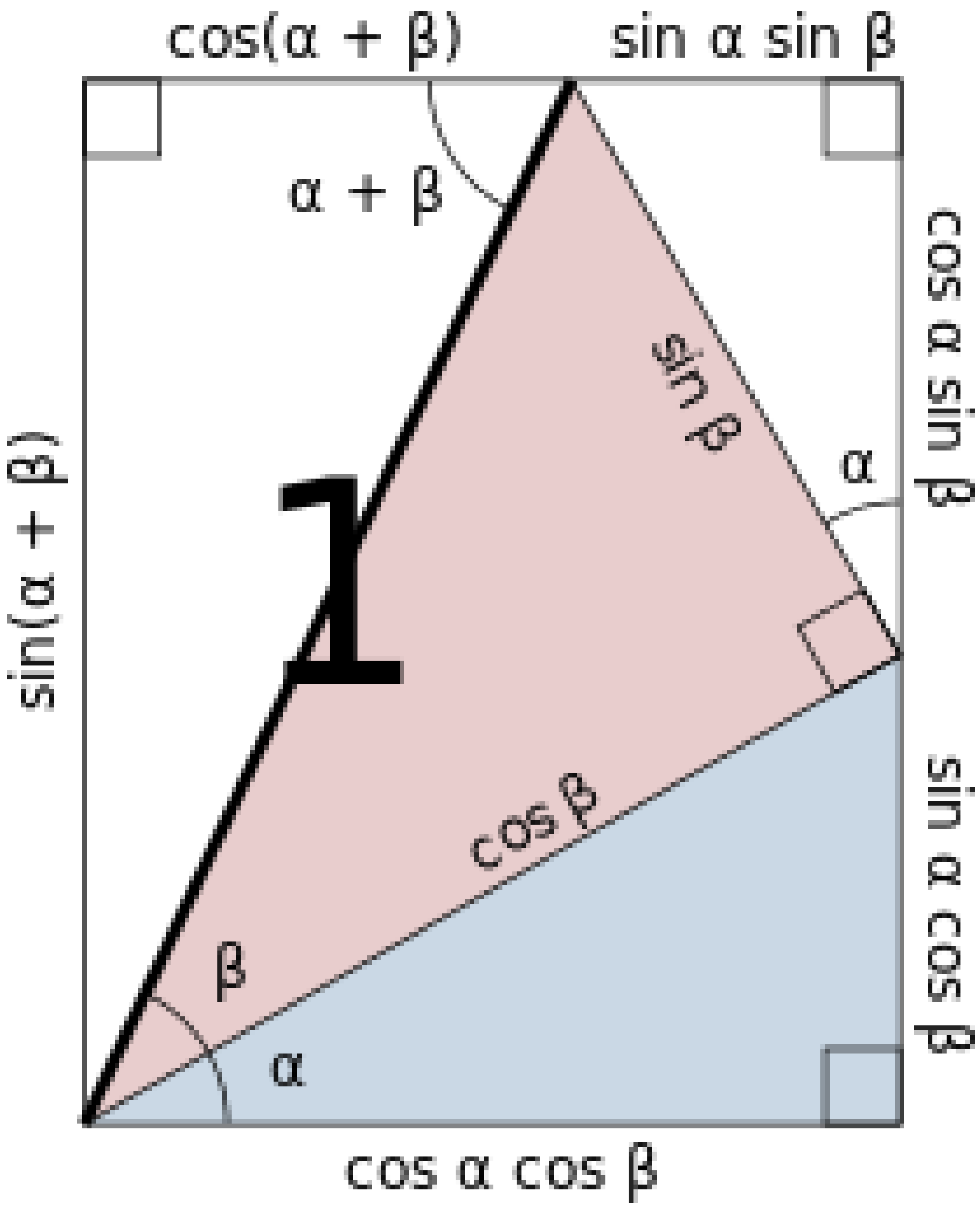
Separating Axis Theorem

- SFML only takes note of a polygon's position, rotation, and point coordinates in local space
- However, using the SAT for an overlap test requires the points to be in world space coordinates
  - Apply rotation then position
  - Note: SAT works with convex shapes only

2-D Rotation

- Given a unit vector (x, y)
  - $x = \cos\theta$
  - $y = \sin\theta$
- Rotating it by α degrees around the origin means new point (x', y')
  - $x' = \cos(\theta + \alpha)$
  - $y' = \sin(\theta + \alpha)$

2-D Rotation



https://en.wikipedia.org/wiki/File:AngleAdditionDiagramSine.svg

2-D Rotation

- Back to the new point (x', y')
- $x' = \cos\theta \cos\alpha - \sin\theta \sin\alpha$

  $= x \cos\alpha - y \sin\alpha$
- $y' = \sin\theta \cos\alpha + \cos\theta \sin\alpha$

  $= y \cos\alpha + x \sin\alpha$

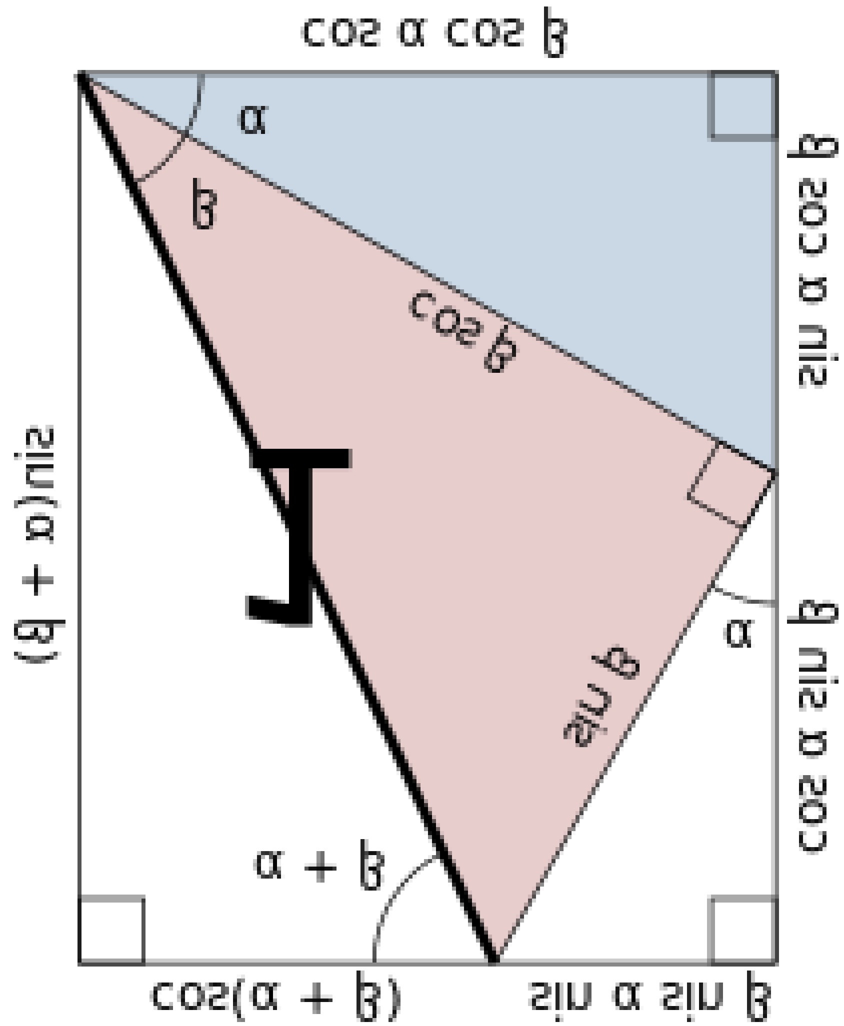
2-D Rotation

- But SFML uses screen coordinates
- Shouldn’t we negate y?
  - –y = sin O
  - –y' = sin (O + M)

2-D Rotation

- Rotation in SFML is clockwise
- You shouldn't negate y!
  - ~~–y = sin O~~
  - ~~–y' = sin (O + M)~~

```
aShape.rotate( deg );
```



https://en.wikipedia.org/wiki/File:AngleAdditionDiagramSine.svg

2-D Translation

- After applying rotation, you can now apply position
  - Which is just a simple addition operation

Edge Normal

- Assuming you defined your polygon in CCW order, an edge's normal is obtained by:
  - First getting the vector that points from one point to the next (the edge itself)
  - Then getting the vector that is CW perpendicular to it (the normal)

Edge Normal

- However, the SAT overlap test will still work even if the normal is facing the wrong way
  - Towards the inside of the polygon and not away from it

Projection

- Now you just need to project all the points of each polygon on the edge normal
  - The end result should be two line segments along the edge normal
- Check if they overlap
  - If they do, repeat the test for the next edge or conclude that the polygons overlap if all edges have already been tested
  - If they do not, the polygons do not overlap

Overlap

- How do you check if the line segments overlap?
  - Hint: Minimum definition of line segment simply requires 2 points
  - Hint#2: All points are being projected on the same normal
  - Hint#3: Check slide set #5

Homework

- Create a program that creates polygons based on text received via standard input
  - All input separated by whitespace characters
  - Input begins with a positive integer N indicating how many shapes to create
  - Input for N shapes follows
  - Note: The program does not check if the shapes are convex

Homework

- For each shape:
  - Input starts with a positive integer P indicating how many points the shape has
  - Input for P+1 points follows
- For each point:
  - Input consists of 2 numbers – a point's local coordinates (x then y)
  - The last "point" is the shape's position (also x then y)

Homework

- There should be a button to toggle rotation on and off
  - Off at the start
  - When toggled on, each shape should rotate at a constant speed
  - But the speed for each shape should be different
- The first polygon should also be movable using the WASD keys

Homework

- Your program should render the convex polygons in one of three different colors:
  - First color if polygon is actually intersecting with another polygon
  - Second color if polygon's AABB is intersecting with another polygon's AABB but polygons themselves are not intersecting
  - Third color if polygons' AABBs do not intersect