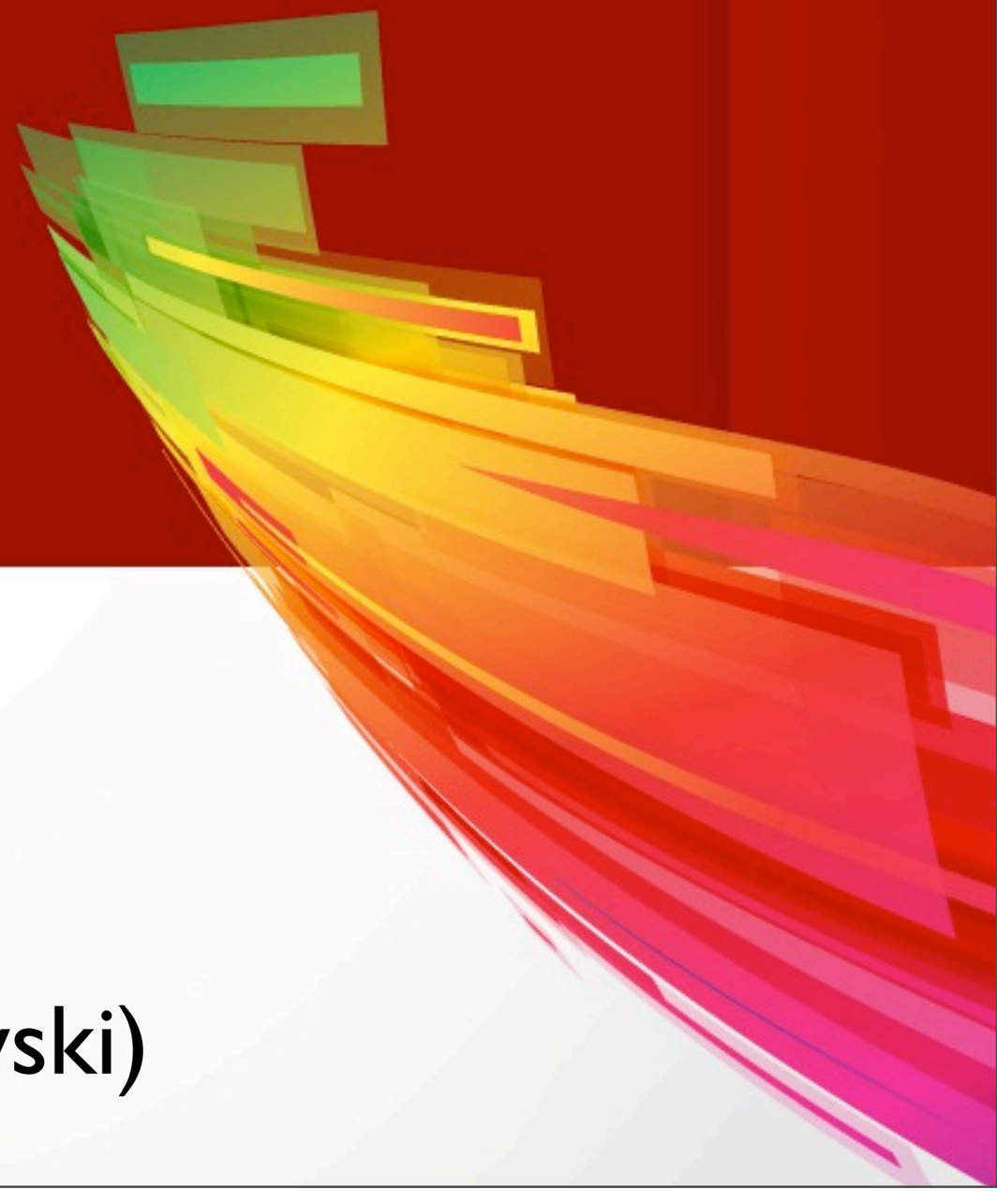
// DEV meetings.pl

Collision Detection



By DevMeetings (Marek Pawłowski)

// DEV meetings.pl

Javascript

Overview

- weak(duck) typing
- object-oriented
- scripting language
- first-class functions
- prototype-base inheritance model
- also considered as functional language (closure and higher-order functions)
- dynamic

Types

- undefined
- Number
- String
- Function
- Object
- Array
- RegExp
- null
- Infinity, -Infinity
- NaN
- Error

Types

- var a; a = undefined;
- a = 1;
- a = “asdf”;
- a = function () {}; function a() {};
- a = new Object(); new function () {}; {};
- a = new Array(10); new Array(10, 9, 8); [10, 9, 8]
- a = new RegExp(‘/^asdf$/’, ‘gi’); /^asdf$/gi;
- a = null;
- a = 1+Number.MAX_VALUE; 1+1.7976931348623157e+308; Infinity
- a = parseInt(‘asdf1’, 10); NaN
- a = new Error(“message”);

// DEV meetings.pl

first-class functions, higher-order functions

- var a = function (arg1, arg2) { return arg1(arg2) };
- a(function (b) { return b*10}, 3);

Creating objects

```
function Animal() {
    this.property = “value”;
    this.sound = function () {...};
}

var a = new Animal();
```

Creating objects

```
function Animal() {}

Animal.prototype.property = “value”;
Animal.prototype.sound = function () {...};

var a = new Animal();
```

Creating objects

```
var Animal = function () {};
function Animal() {}
```

```
Animal.prototype = {
  property: “value”,
  sound: function () {...}
};

var a = new Animal();
```

Creating objects

```
var Animal = {
    property: “value”,
    sound: function () {...}
};

var a = Object.create(Animal);
```

Public, private and protected

- There is not such thing like this in JS ;-(
- This is very big lack in terms of modularity
- But there are few way how to solve/ emulate this

Public, private and protected

```
// By clousure http://www.crockford.com/javascript/private.html
function Container(param) {
    this.member = param; // public
    this.a = function () {}; // public

    var secret = 3; // private
    var b = function b() {}; // private

    this.c = function () { //privileged
        alert(secret);
     };
}
Container.prototype.d = function () {}; //public
```

Public, private and protected

```
// By code convention
function Container() {};
Container.prototype = {
    __a: 1, // private
    _b: 2, // protected
    c: 3,  // public

    __d: function () {}, // private
    _e: function () {},  // protected
    f: function () {}    // public
};
```

// DEV meetings.pl

prototype-base inheritance model

```
function Animal() {};
Animal.prototype = {
  sound: function () {...}
};

function Cat() {};
Cat.prototype = new Animal();

var c = new Cat();
```

prototype-base inheritance model

```
function Animal() {
    this.weight = “10kg”;
}
...
function Cat() {
    Animal.apply(this, arguments);
};
Cat.prototype = new Animal();

var c = new Cat();
```

// DEV meetings.pl

prototype-base inheritance model

```
...
Cat.prototype = Animal.prototype;
...
Cat.prototype.sound = null;
a.sound(); // ???
...
```

// DEV meetings.pl

prototype-base inheritance model

```
function Animal() {
    this.weight = “10kg”;
}
...
function Cat() {
    Animal.apply(this, arguments);
}
var temp = function () {};
temp.prototype = Animal.prototype;
Cat.prototype   = new temp();

var c = new Cat();
```

prototype-base inheritance model

```
...
var temp = function () {};
temp.prototype = Animal.prototype;
Cat.prototype   = new temp();
Cat.prototype.constructor = Cat;

var c = new Cat();
```

prototype-base inheritance model

```
function inherits (Child, Parent) {
    var temp = function () {};
    temp.prototype = Parent.prototype;
    Child.prototype = new temp();
    Child.prototype.constructor = Child;
}
```

prototype-base inheritance model

```
if (typeof Object.create !== “function” ) {
 Object.create = function (o) {
    var temp = function () {};
    temp.prototype = o;
    Child.prototype = new temp();
    return= Child;
 };
}
```

prototype-base inheritance model

```
function Animal() {
    this.weight = “10kg”;
};

...
function Cat() {
    Animal.apply(this, arguments);
};

inherits(Cat, Animal);

var c = new Cat();
```

// DEV meetings.pl

prototype-base inheritance model

```
var Animal = {
    weight: “10kg”,
    sound: function () {}
};
var Cat = Object.create(Animal);
Cat.weight = “25kg”;

var c = Object.create(Cat);
c.name = “garfield”;
```

Dynamic

```
Function.prototype.memoize = function () {
    var memo = {}, that = this;
    var fn       =  function () {
        var args = Array.prototype.slice.call(arguments);
        var item = memo[args];

        return item || (item = that.apply(null, args));
    };

    fn.unmemoize = function () {
        return that;
    };
    return fn;
};
Function.prototype.unmemoize = function () {
    throw new Error("This function wasn't memoize before!!!");
};
```

Dynamic

```
var fib = function fib(n) {
    if (n === 0 || n === 1) {
        return 1;
    }
    iteration++;
    return fib(n - 1) + fib(n - 2);
};

fib(45); // 1 836 311 902 WTF!!!
fib = fib.memoize();
fib(45);// 44
```

// DEV meetings.pl

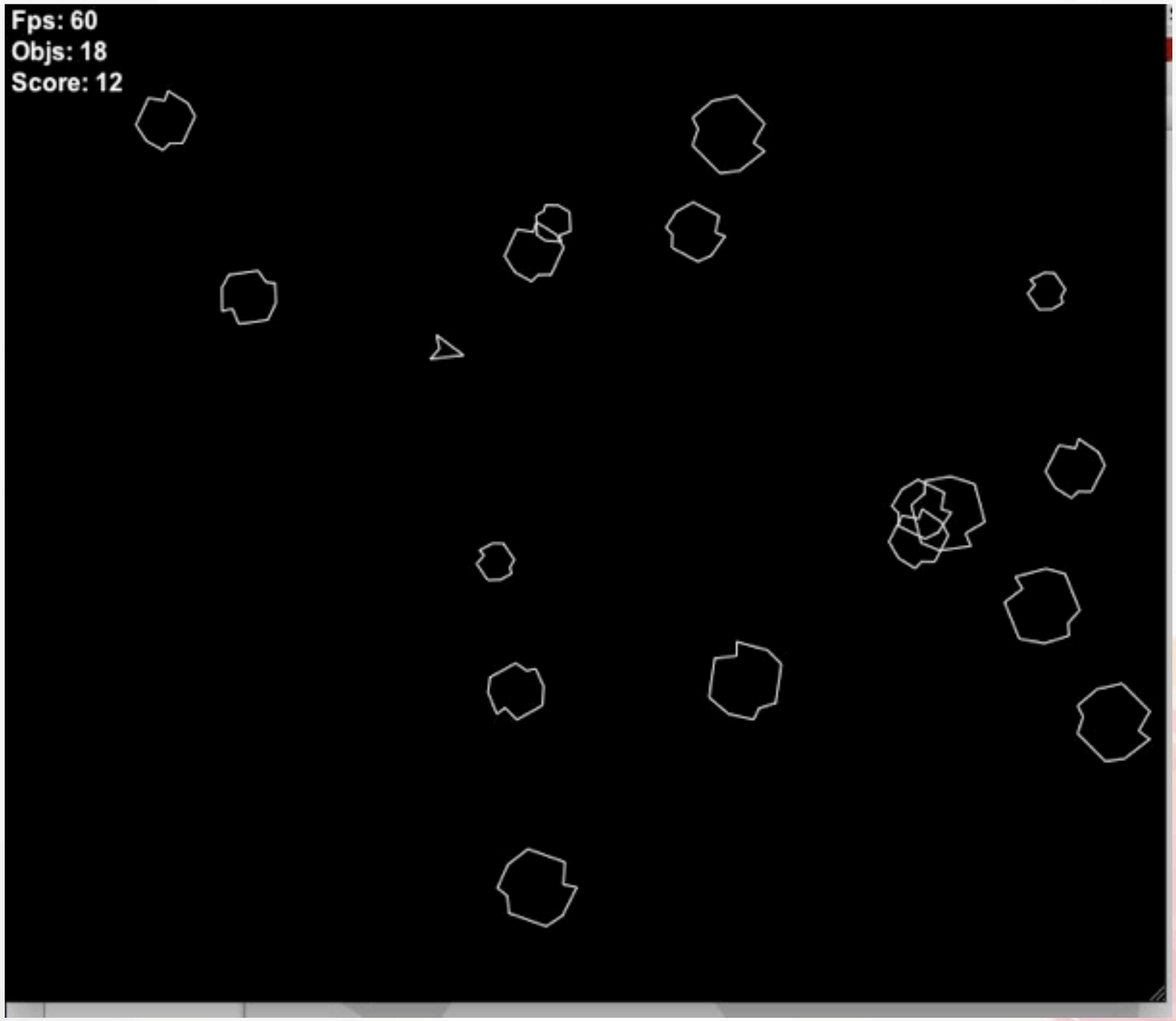
Let's have some fun!

Stwórz obiekt aspect z następującymi metodami:
- createAspect, która ma za zadanie podpiąć funkcję aspectFn pod wykonanie metody fnName w obiekcie obj. Metoda przyjmuje następujące parametry:
 obj – referencja do obiektu (w przypadku null/undefined – odwołanie do obiektu globalnego)
 fnName – nazwa funkcji (string)
 aspectFn – funkcja 'aspektowa'
 when – string 'after' lub 'before' (domyślnie 'before')
 once – boolean; jeśli true aspekt wykona się tylko raz (domyślnie false)

- removeAspect – usuwa wcześniej zadeklarowany aspekt; parametry wejściowe: obj, fnName, aspectFn, when

Jeśli fnName nie jest metodąobiektu obj, metody rzucają TypeError.
fnName może być tablicą funkcji bądź wyrażeniem regularnym, np. /get.+/

Asteroids

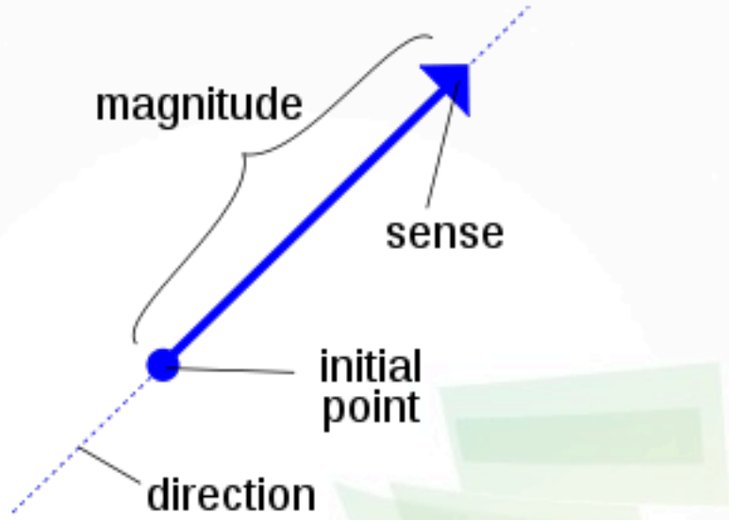


// DEV meetings.pl

What we need more than js knowledge

- Simple mathematics
- Simple physics
- Game models
- Visualization
- Collision detection

Vector



- “A Euclidean vector is frequently represented by a line segment with a definite direction, or graphically as an arrow, connecting an initial point A with a terminal point B,[3] and denoted by AB” - wikipedia

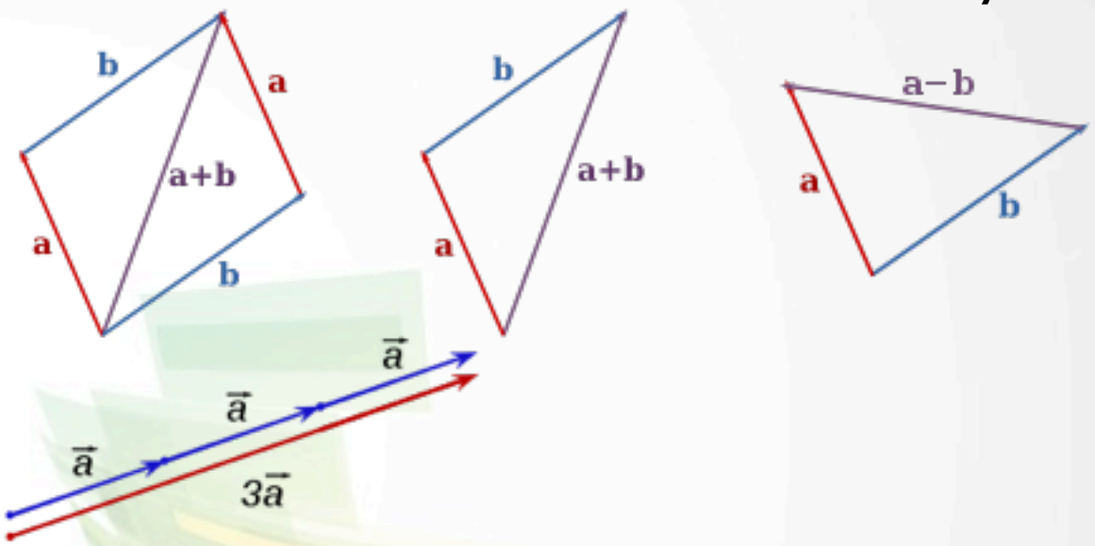
// DEV meetings.pl

Vector Arithmetic

$$\mathbf{e}_1 = (1,0,0),\ \mathbf{e}_2 = (0,1,0),\ \mathbf{e}_3 = (0,0,1)$$

basic vectors in Cartesian coordinate system

$$\mathbf{a} + \mathbf{b} = (a_1 + b_1)\mathbf{e_1} + (a_2 + b_2)\mathbf{e_2} + (a_3 + b_3)\mathbf{e_3}.$$
$$\mathbf{a} - \mathbf{b} = (a_1 - b_1)\mathbf{e_1} + (a_2 - b_2)\mathbf{e_2} + (a_3 - b_3)\mathbf{e_3}.$$

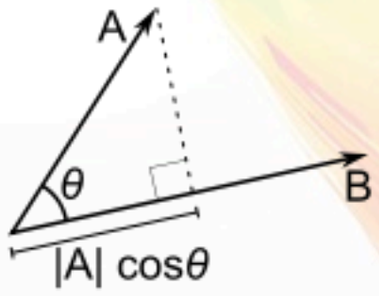


$$r\mathbf{a} = (ra_1)\mathbf{e_1} + (ra_2)\mathbf{e_2} + (ra_3)\mathbf{e_3}.$$

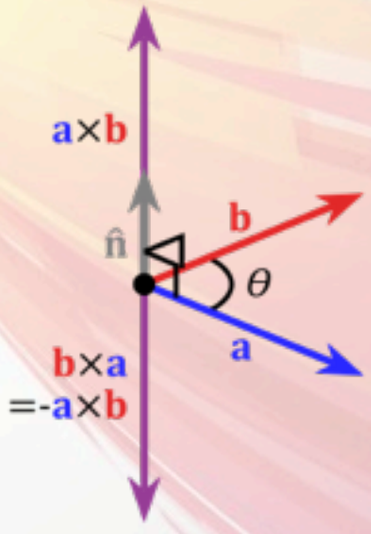
$$\|\mathbf{a}\| = \sqrt{{a_1}^2 + {a_2}^2 + {a_3}^2}$$

length/magnitude

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|\,\|\mathbf{b}\| \cos\theta$$



$$\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\|\,\|\mathbf{b}\| \sin(\theta)\,\mathbf{n}$$

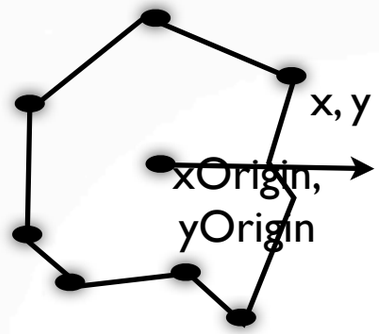


glMatrix

- http://code.google.com/p/glmatrix/wiki/Usage

// DEV meetings.pl

Simple physics - movement




```
var Shape = {
  position: [1, 1, 0], // vec3
  vertices: [...],      // vec3 []
  velocity: [1,1, 1]  // vec3
};
```

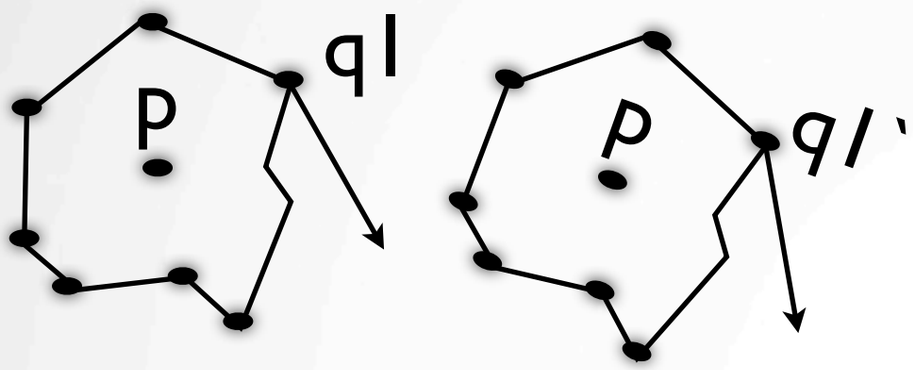
// DEV meetings.pl

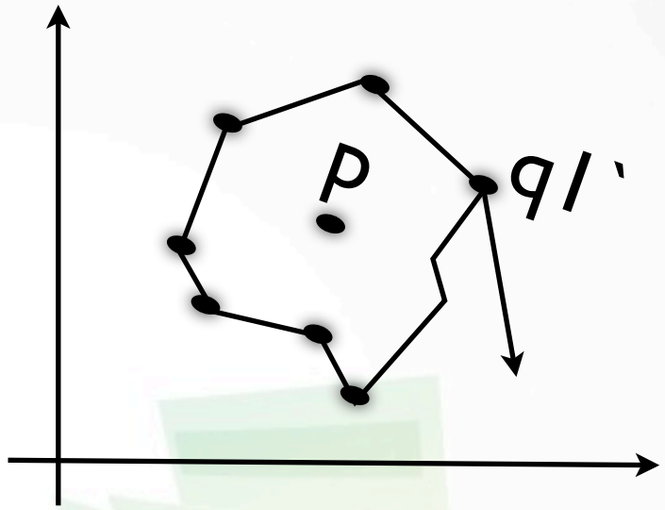
Simple physics - movement

```
var Shape.prototype.integrate = function (time) {
    vec3.add(position,
        vec3.scale(velocity, time, vec3.create()));

    // if acceleration exists add acceleration to velocity
};
```

Simple physics - rotation



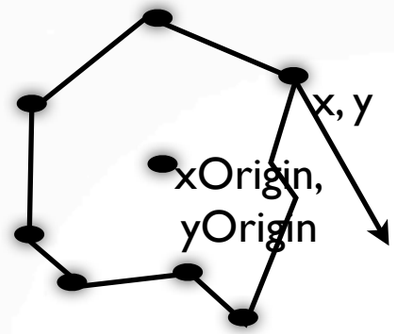


$$ql` = \begin{vmatrix} cosA & -sinA \\ sinA & cosA \end{vmatrix} ql$$

$$ql` = \begin{vmatrix} cosA & -sinA \\ sinA & cosA \end{vmatrix} ql + p$$

$$ql` = \begin{vmatrix} cosA & -sinA \\ sinA & cosA \end{vmatrix} \begin{vmatrix} xql \\ yql \end{vmatrix} + \begin{vmatrix} xp \\ yp \end{vmatrix}$$

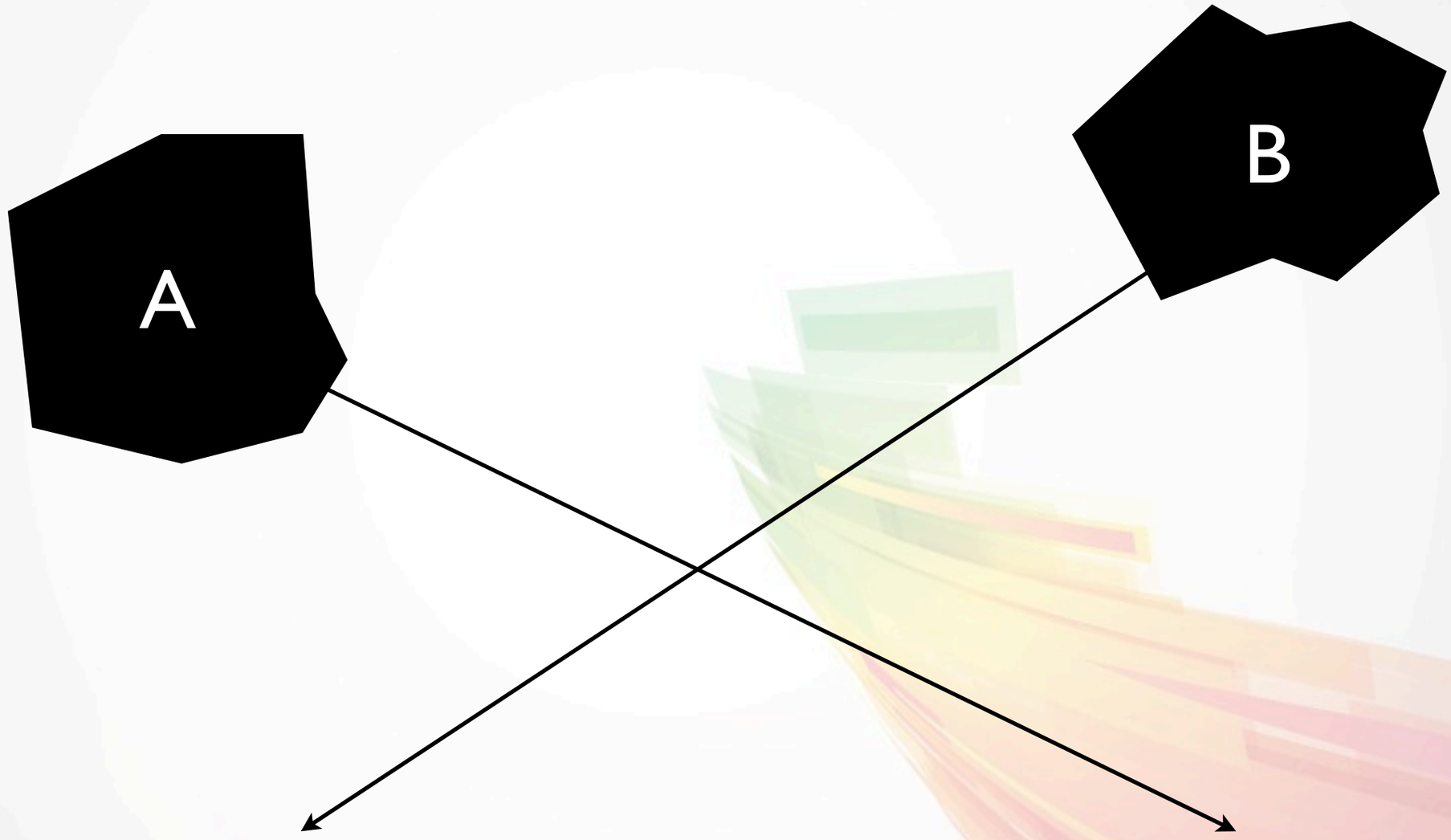
Simple physics - rotation



- x = (cos * x) - (sin * y);
- y = (sin * x) + (cos * y);
  - sin = sin(angle * PI/180);
  - cos = cos(angle * PI/180);

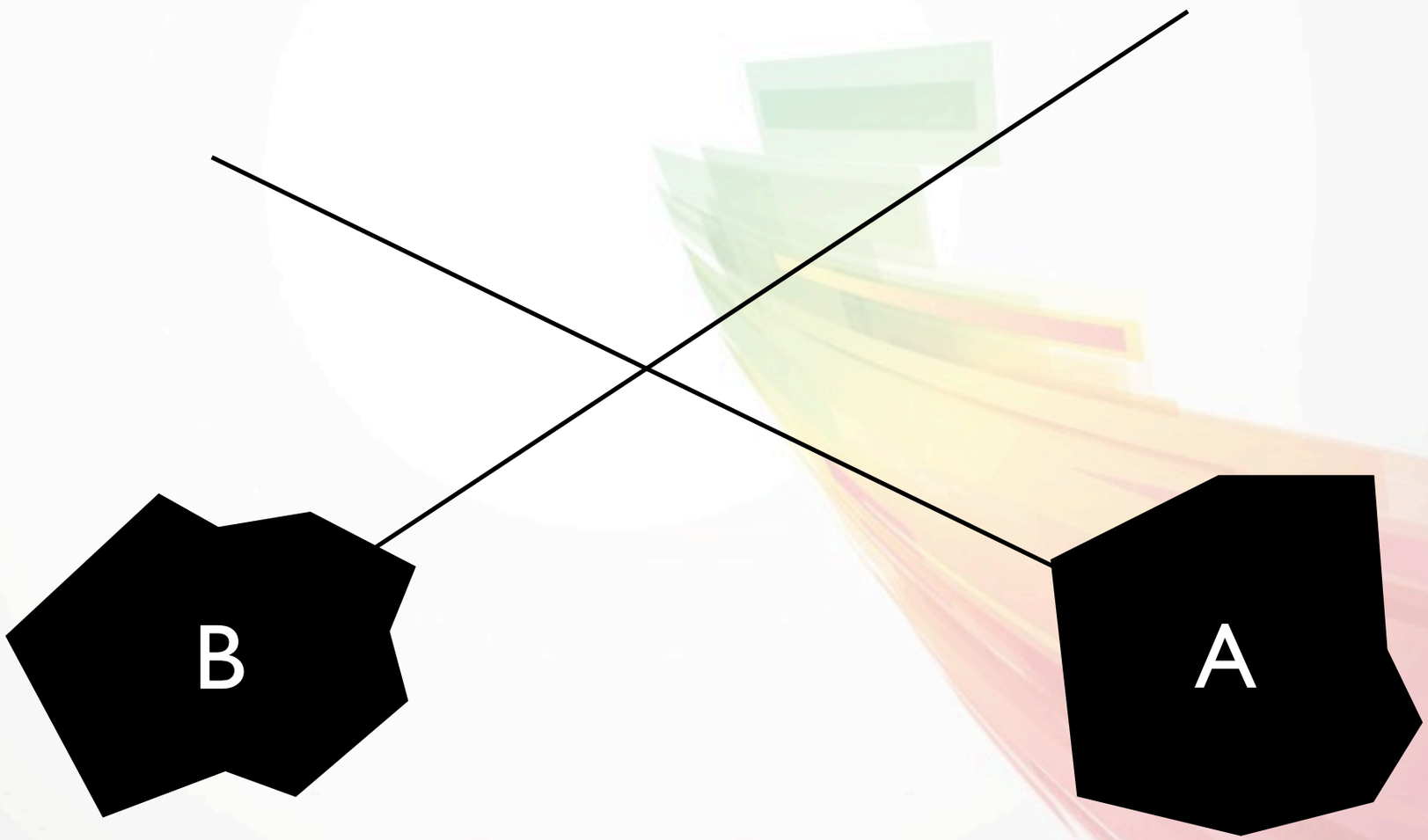
// DEV meetings.pl

Sequential vs Simultaneous Motion



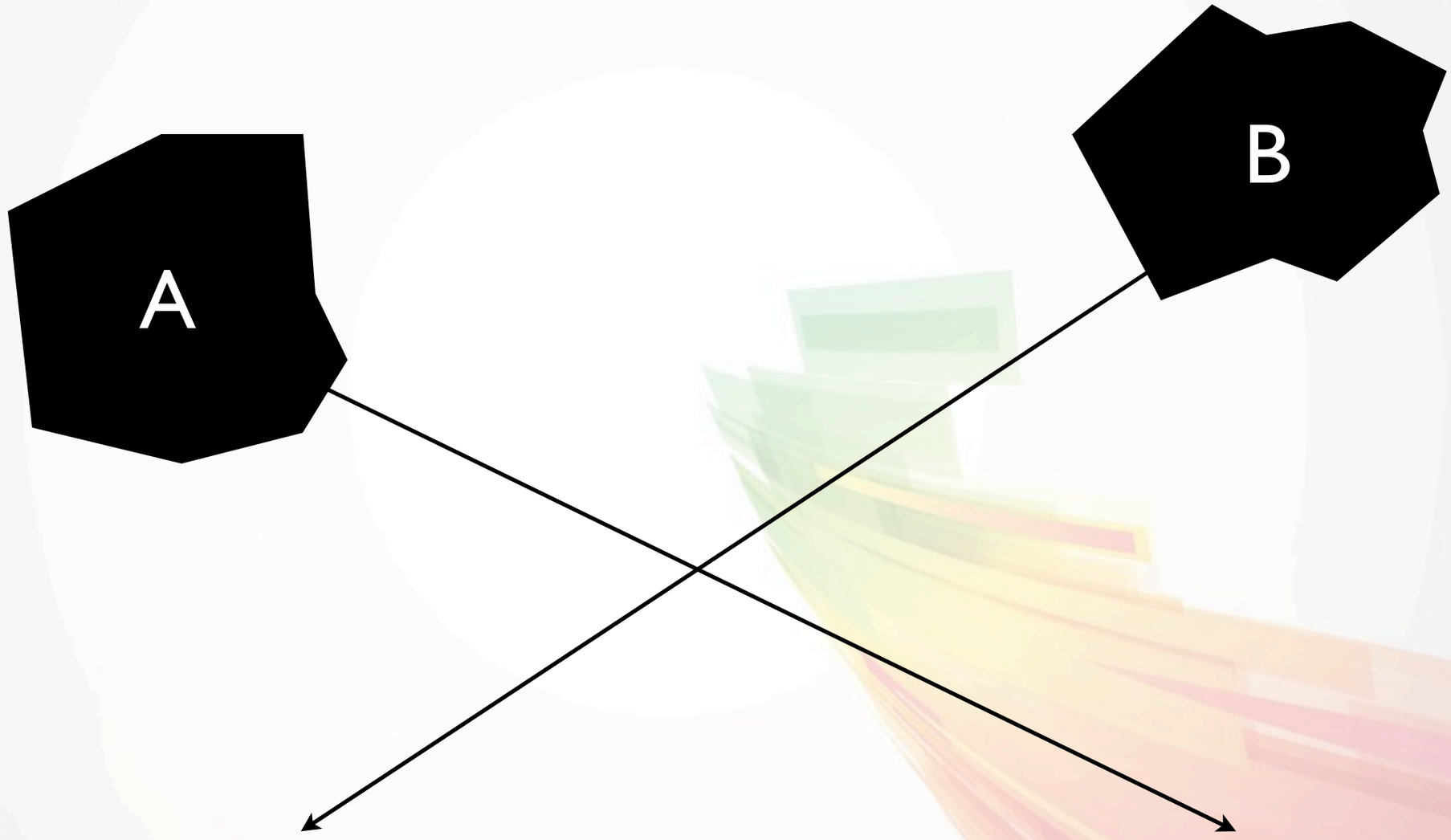
// DEV meetings.pl

Sequential vs Simultaneous Motion



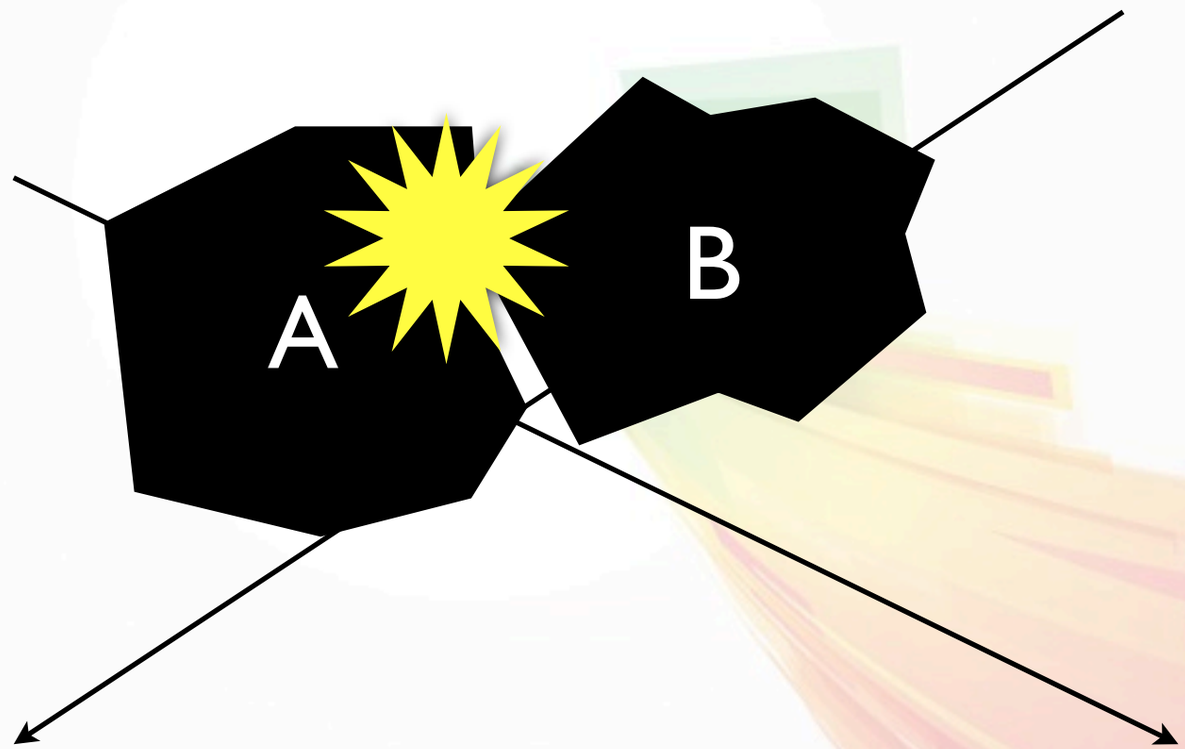
// DEV meetings.pl

Sequential vs Simultaneous Motion



// DEV meetings.pl

Sequential vs Simultaneous Motion



Sequential motion

- Objects are moved one object at a time
- Collisions are detected and resolved before the process continues with next object

Collision detection

- broad phase
- narrow phase

Visualization

- Canvas
- mozRequestAnimationFrom, webkitRequestAnimationFrame

// DEV meetings.pl

Miniframework - core prototype

```
app.core.Object.prototype = {
    getDocument: function () {...},
    getWindow: function () {...},
    addListener: function (name, fn, ctx /* args */) {...},
    existEvent: function (name) {...},
    fireEvent: function (name) {...},
    fireDataEvent: function (name, data) {...}
};
```

// DEV meetings.pl

Miniframework - skeleton

```
app.core.Object.define("app.namespace.Object", {
    extend: app.core.Object,
    constructor: function () {
        //call parent constructor
        arguments.callee.prototype.uper.apply(this, arguments);

    },
    static: {},
    member: {}
});
```

Miniframework - model

```
app.core.Object.define("app.model.Shape", {
    ...
    member: {
        setX: function (newValue) {
            this.__position[0] = newValue;
        }
    }
});
```

Miniframework - view

```
app.core.Object.define("app.view.Shape", {
    constructor: function (model) {
        ...
        this.__model = model;
    },
    member: {
        render: function (ctx) {
            // iterate over all vertices from model and draw
            // lines between them
        }
    }
});
```

// DEV meetings.pl

Miniframework - controller

```
app.core.Object.define("app.controller.Character", {
    ...
    constructor: function (model, view) {...},
    member: {
        dispatch: function (event) {
            if (event) {
                switch (event[1]) {
                    case app.event.Object.LEFT:
                        this.__model.changeDirection(5);
                        break;
                    ...
                    case app.event.Object.UP:
                        this.__model.increaseVelocity();
                        break;
                }
            }
        }
    }
});
```

TASK

- Create moving objects on the stage using given mathematics

// DEV meetings.pl

Bounding Volumes

Motivation

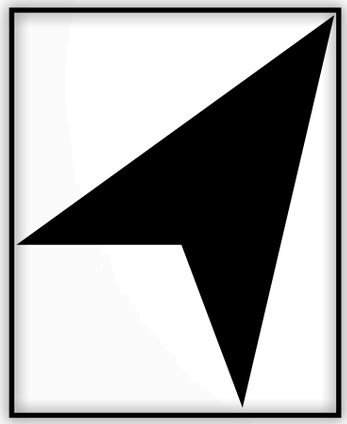
- Simplify objects
- Minimize expensive test between complex object
- Fast rejection objects.

// DEV meetings.pl

Types of bounding volumes

Sphere

AABB



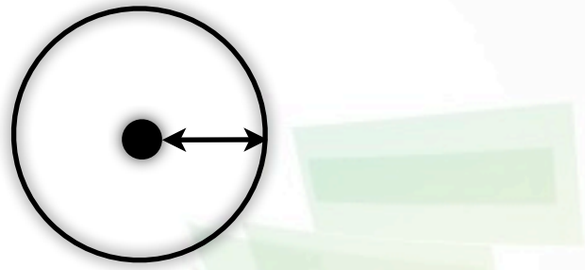
OBB

8-DOP

Sphere

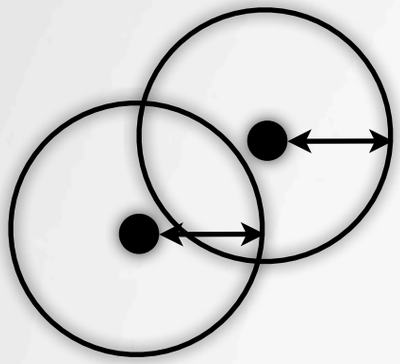
- Inexpensive intersection test
- Rotationally invariant
- Expensive computation

Representation



```
var Shpere = {
  c: [1, 1], // vec3
  r: 1,
};
```

Intersection



```
function testSphereSphere (a, b) {
  var d = vec3.subtract(a.c, b.c, vec3.create());
  var distance2 = vec3.dot(d, d);
  var radiusSum = a.r + b.r;

  return distance2 <= radiusSum * radiusSum;
}
```

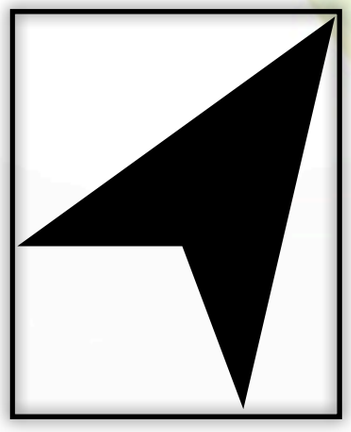
// DEV meetings.pl

Computing (ritterSphere)

- Find most separated points on AABB
- And create sphere from distant points

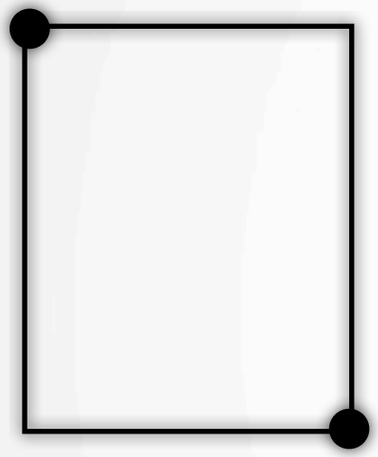
Axis-aligned bounding box

- At all times faces are parallel to the axes of coordinate system
- Fast overlap check (direct comparison of individual coordinate values)



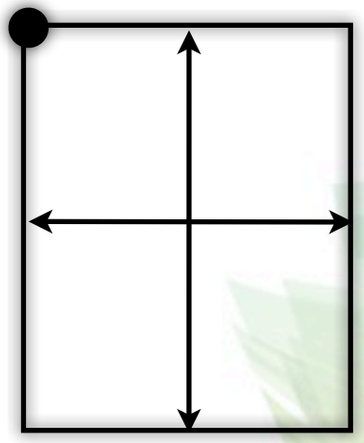
Representation

Min-max point



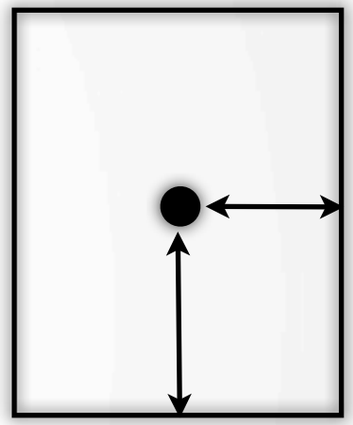
```
var AABB = {
  min: [1, 1], // vec3
  max: [2, 2], // vec3
};
```

Min point-widths



```
var AABB = {
  min: [1, 1], // vec3
  d: [1, 1],
};
```

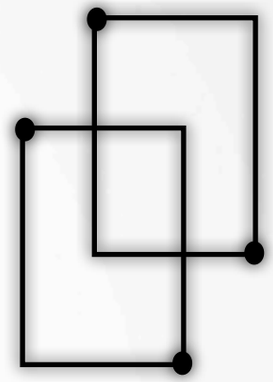
Center-radius



```
var AABB = {
  c: [1, 1], // vec3
  r: [0.5, 0.5],
};
```

// DEV meetings.pl

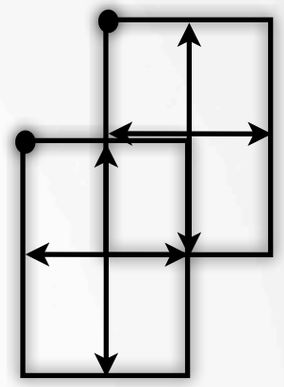
Min-max point - intersection



```
function testAABBAABB (a, b) {
  if (a.max[0] < b.min[0] || a.min[0] > b.max[0])
    return 0;
  if (a.max[1] < b.min[1] || a.min[1] > b.max[1])
    return 0;


  return 1;
};
```

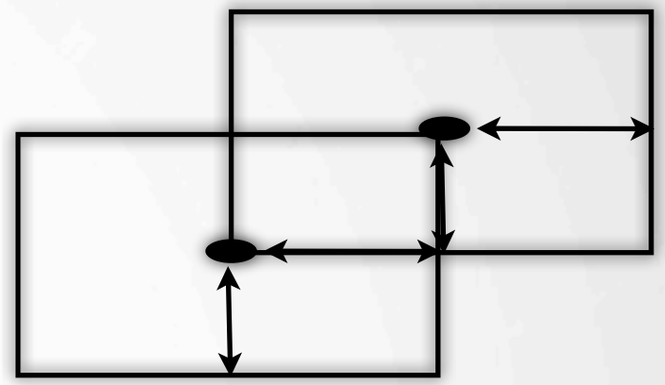
Min-widths - intersection



```
function testAABBAABB (a, b) {
  var t;
  if ((t = a.min[0] - b.min[0]) > b.d[0] || -t > a.d[0])
    return 0;
  if ((t = a.min[1] - b.min [1]) > b.d[1] || -t > a.d[1])
    return 0;


  return 1;
}
```

Center-radius - intersection



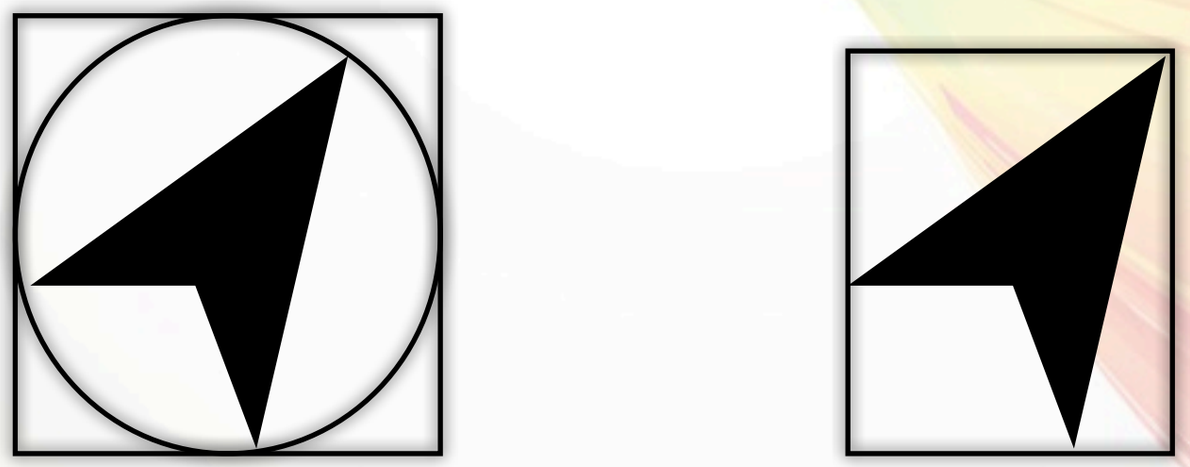
```
function testAABBAABB (a, b) {
  var abs = Math.abs;

  if (abs(a.c[0] - b.c[0]) > (a.r[0] + b.r[0]))
    return 0;
  if (abs(a.c[1] - b.c[1]) > (a.r[1] + b.r[1]))
    return 0;


  return 1;
};
```

Computing

- AABB from the Object Bounding Sphere
- AABB reconstructed from Original Point Set



AABB reconstructed from Original Point Set

- Iterate over all vertices and find min-max points

// DEV meetings.pl

AABB from the Object Bounding Sphere

- Simply take center point and radius for halfwidths

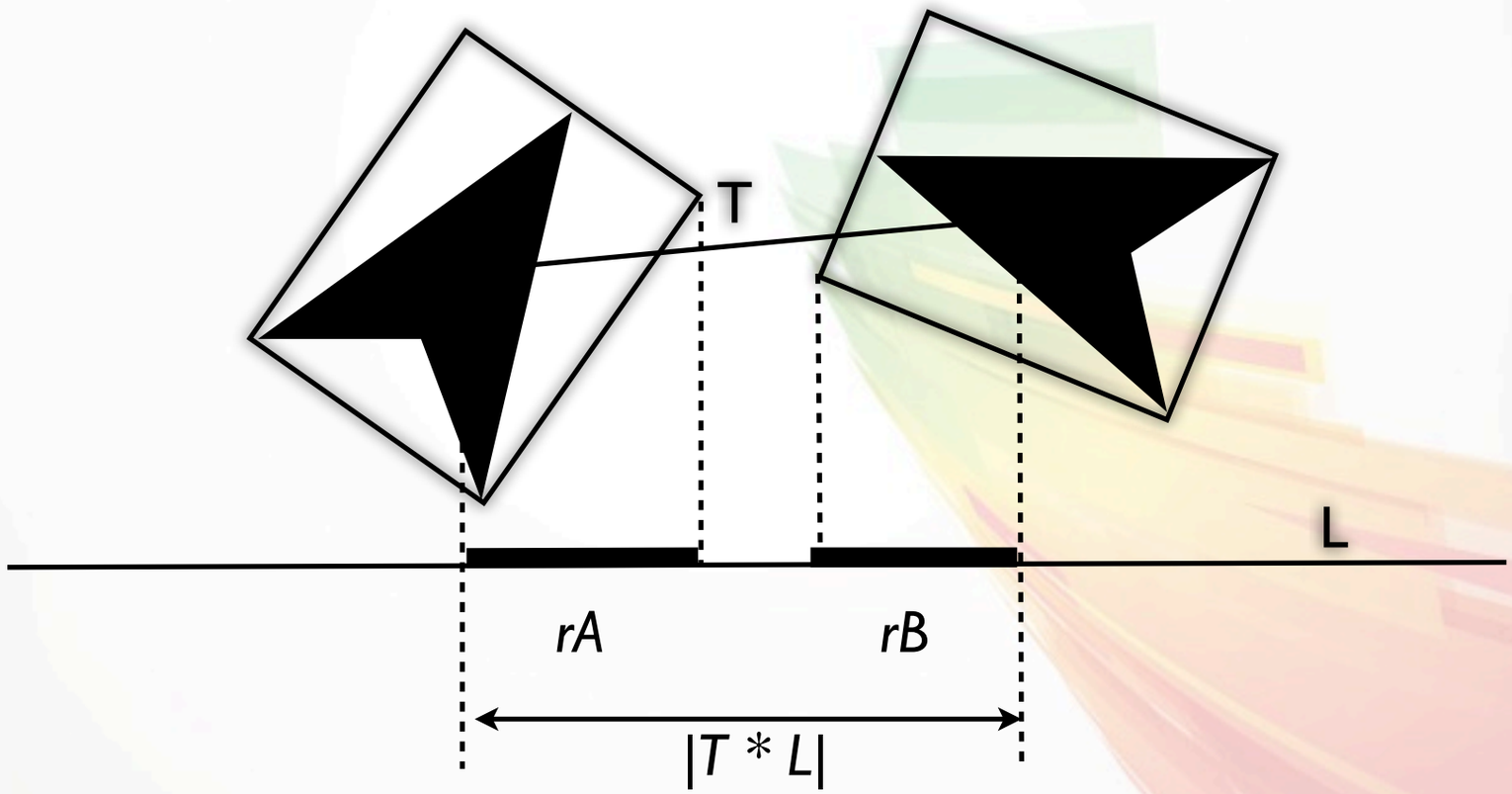
Oriented Bounding Box

- Tight fit
- 15 separating axis needed to determine intersection between OBB-OBB

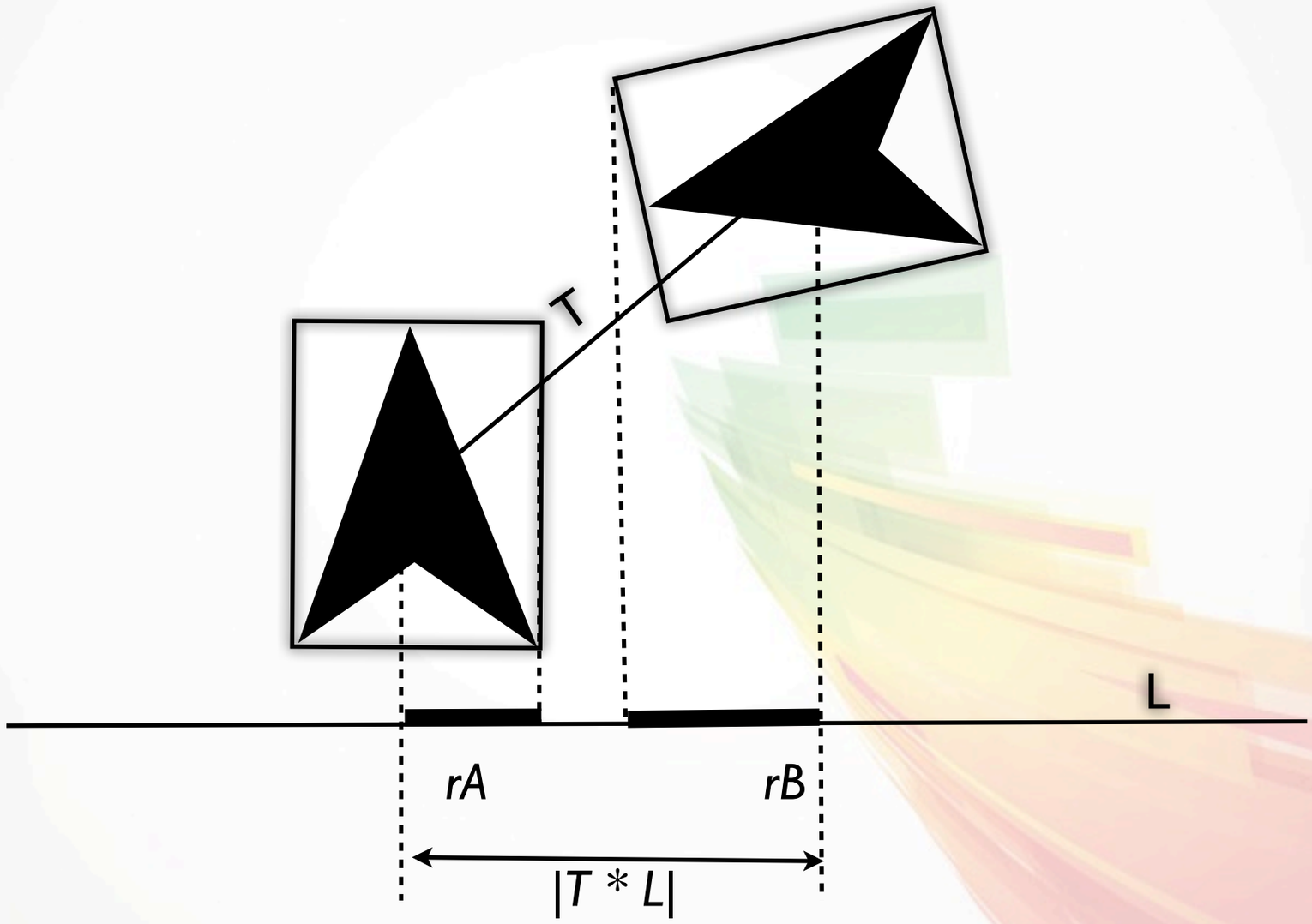
Oriented Bounding Box

```
var OBB = {
  c: [1, 1], // vec3
  u: [vec3, vec3, vec3], //local x,y z-axes
  e: [1, 2] //halfwidths
};
```

Intersection



Intersection



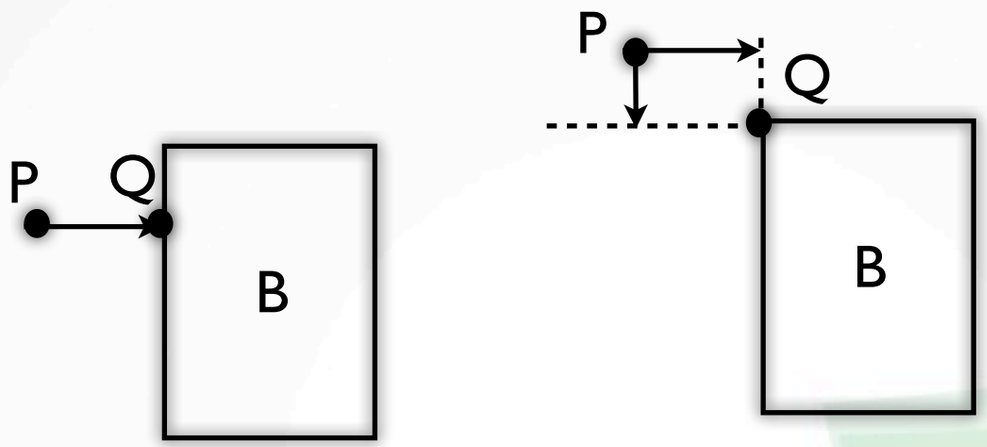
TASK

- Add creation of bounding box to the game
- Add collision check between all object
- Implement collision test:
  - Sphere 2 Sphere
  - 3 types of bounding box

// DEV meetings.pl

Primitive Tests

Closest point on AABB to Point



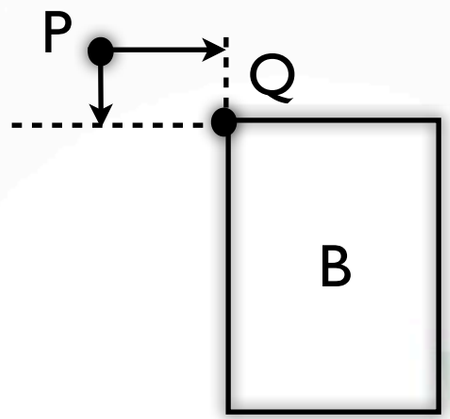
- Iterate over all point axis and find:
  - max(p[i], aabb.min[i])
  - min(p[i], aabb.max[i])
- clamp it to the box or keep it as is

Closest point on AABB to Point

```
function closestPtPointAABB (point, aabb) {
    var q = vec3.create();

    for (var v, i = 0; i < 3; i++) {
        v = p[i];
        if (v < aabb.min[i]) v = aabb.min[i];
        if (v > aabb.max[i]) v = aabb.max[i];
        q[i] = v;
    }
    return q;
};
```

Distance of point to AABB



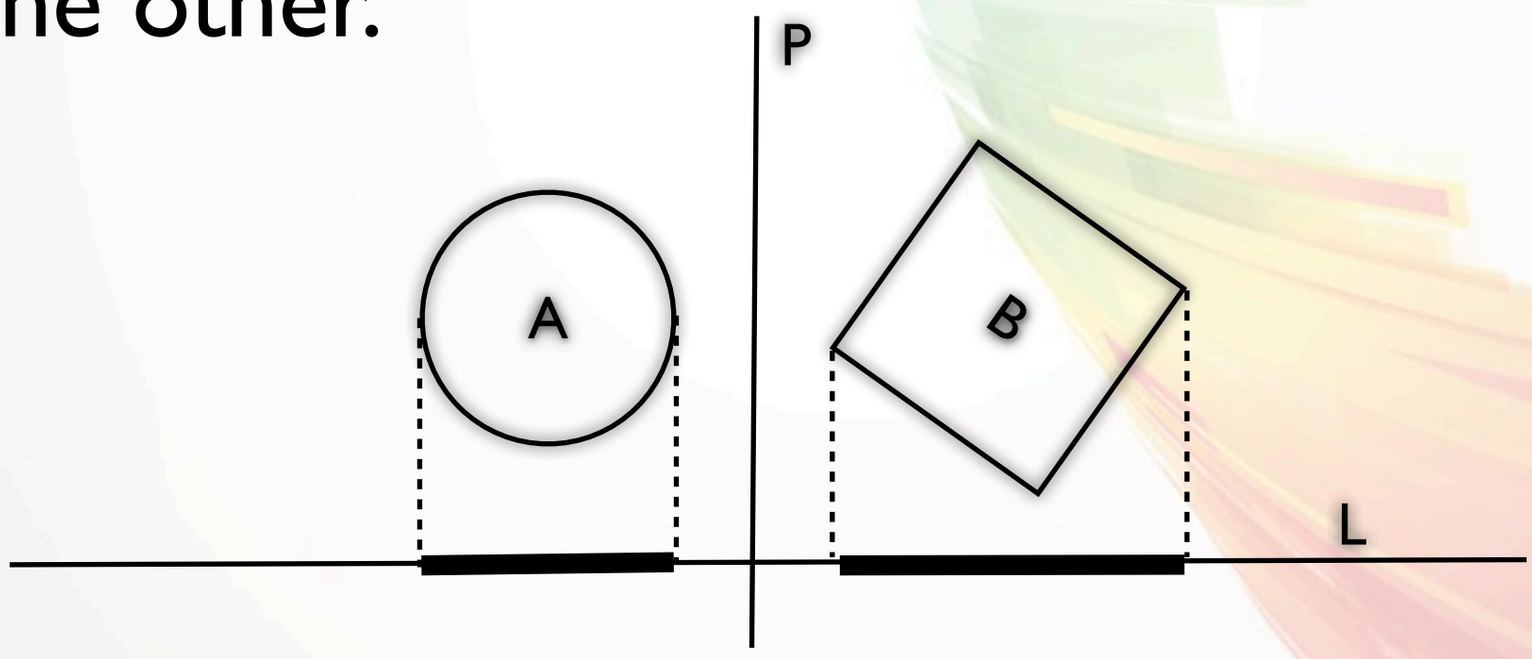
- For each axis count any excess distance outside box extents

Distance of point to AABB

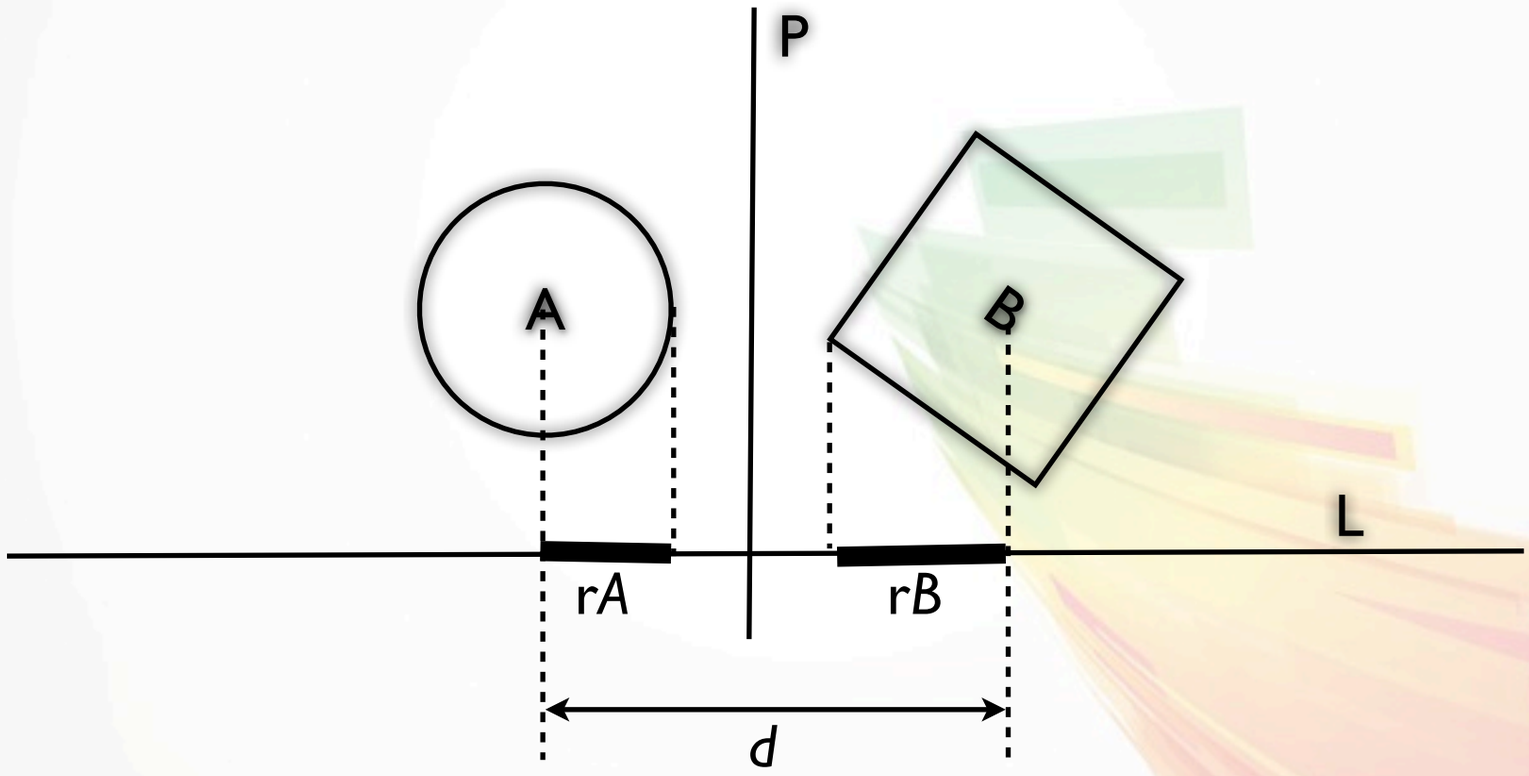
```
function sqDistPointAABB (point, aabb) {
  var sqDist = 0;
  for (var v, i = 0; i < 3; i++) {
    v = point[i];
    if (v < aabb.min[i])
     sqDist += (aabb.min[i] - v) * (aabb.min[i] - v);
    if (v > aabb.max[i])
     sqDist += (v - aabb.max[i]) * (v - aabb.max[i]);
  }
  return sqDist;
}
```

Separating-axis test

- Given 2 **convex** sets A and B, either the two sets are intersecting or there exists a separating hyperplane P such that A is on one side of P and B is on the other.



Separating-axis test



$$rA + rB < d$$

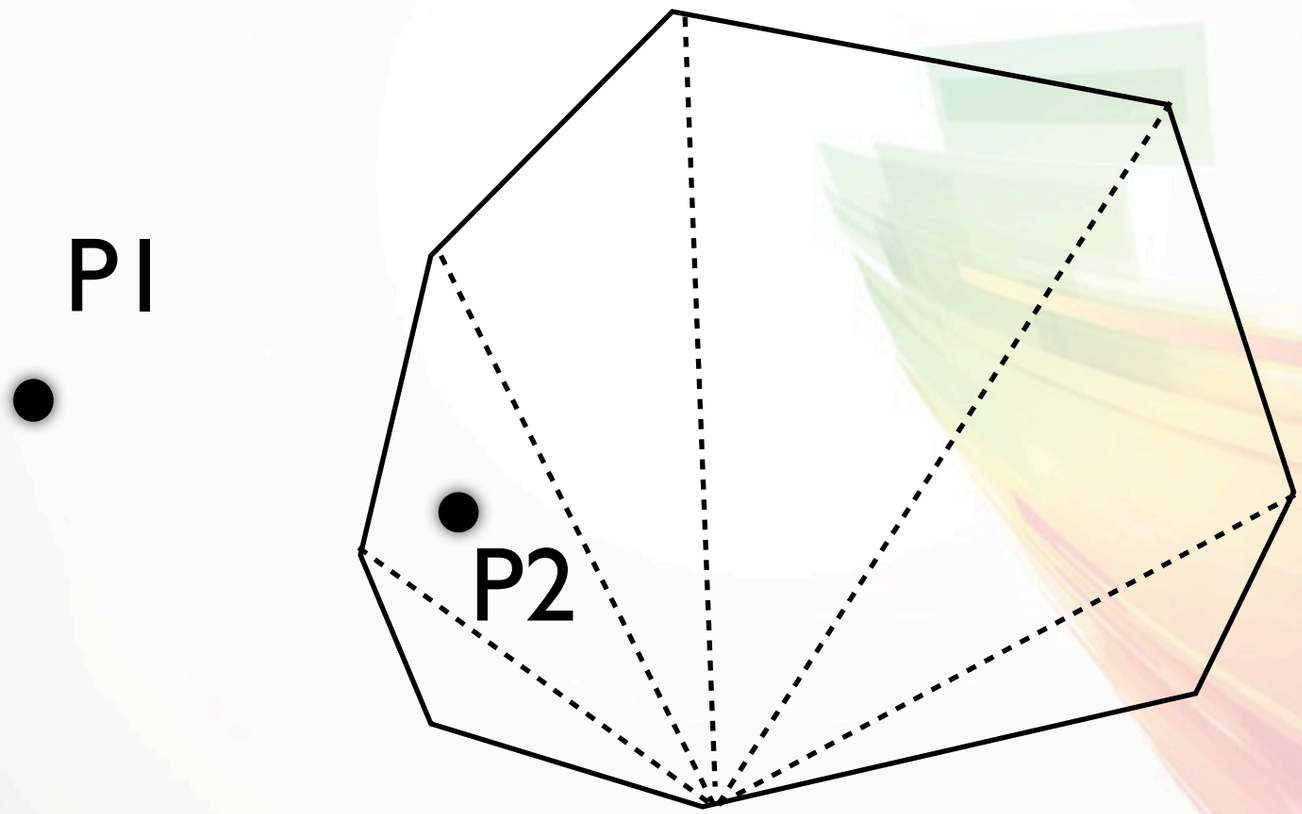
Testing sphere against AABB

```
function testSphereAABB (sphere, aabb) {
  var sqDist = sqDistPointAABB(sphere.c, aabb);

  return sqDist <= sphere.r * sphere.r;
}
```

Testing point in polygon

- Testing if point is in one of triangles inside polygon

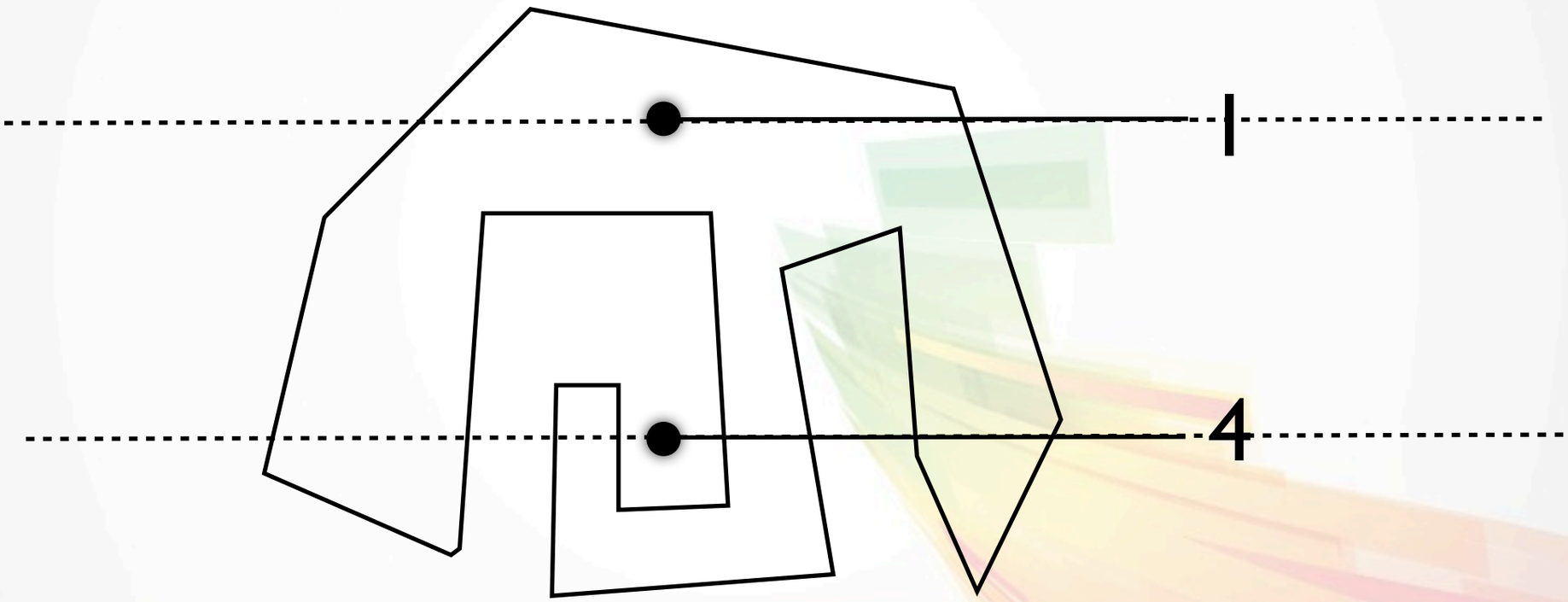


Testing point in Triangle

- Look in to collision.js file

Testing point in polygon

- Shooting rays and count crossing points



Testing point in polygon

- So in our case if we detect possibility of collision we need to test all vertices if they are in other

Task

- Add precise checking for collision between to polygons

// DEV meetings.pl

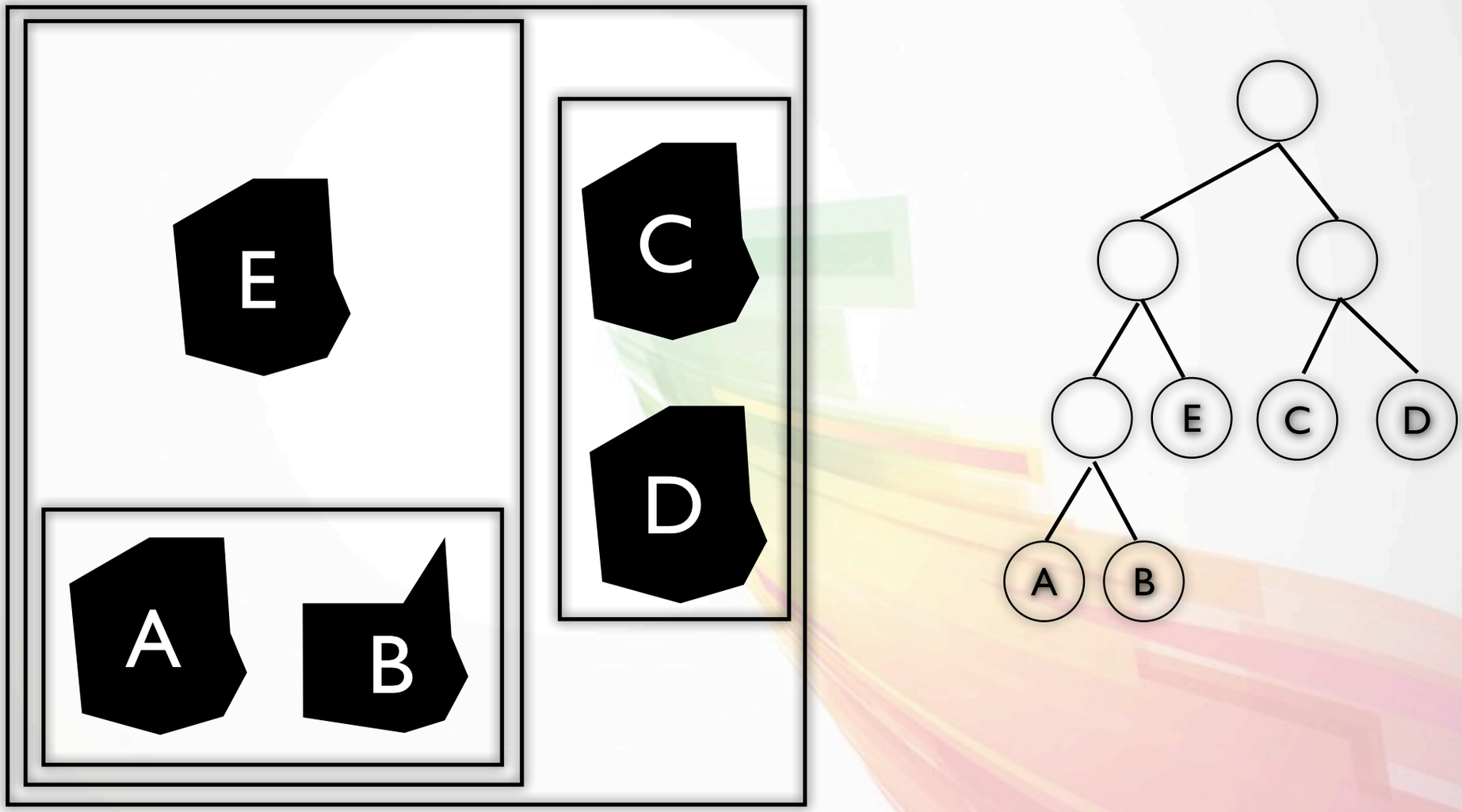
Bounding Volume Hierarchies

Motivation

- Reduce complexity by merging objects

// DEV meetings.pl

Bounding volume hierarchy



// DEV meetings.pl

Spatial Partitioning

Motivation

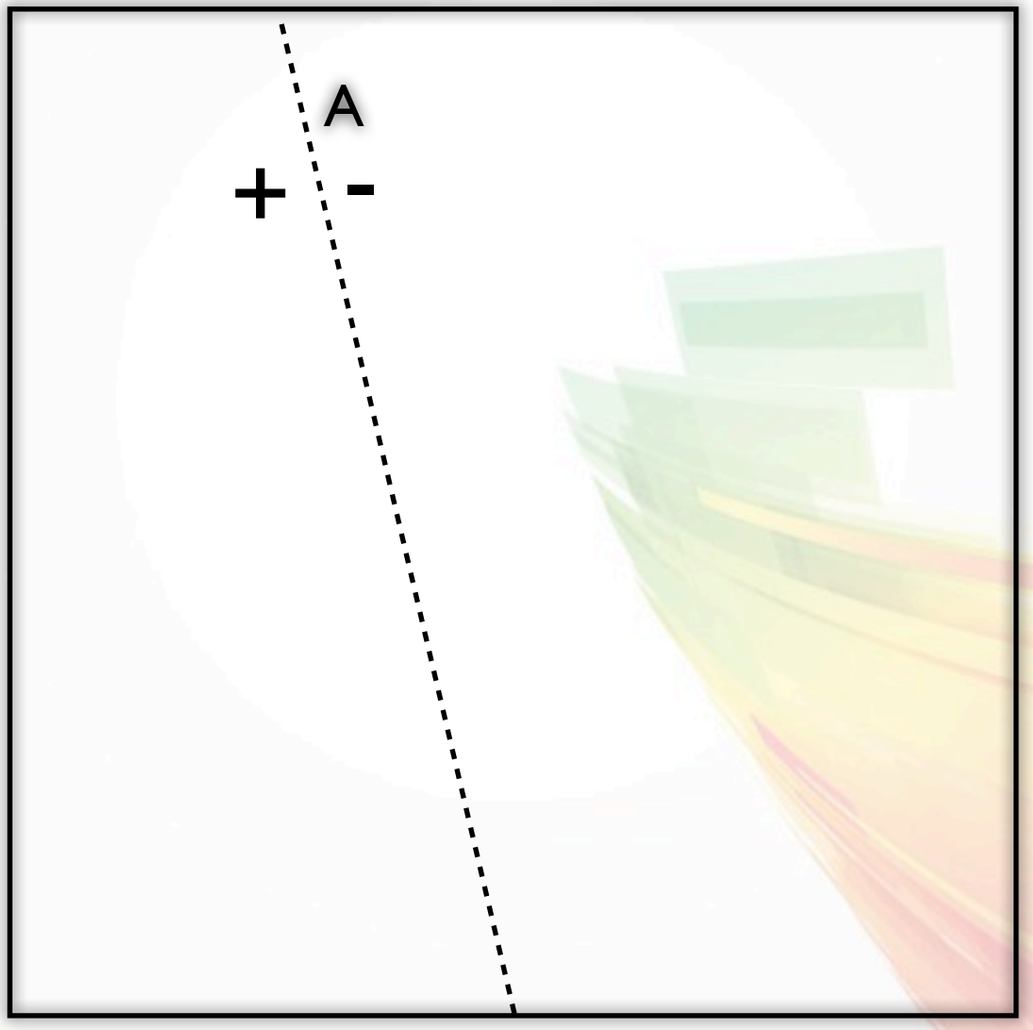
- Nothing is faster than not having to perform a task
- Objects intersect only if they overlap the same region of space
- Test for all elements on the stage cost O(n2)

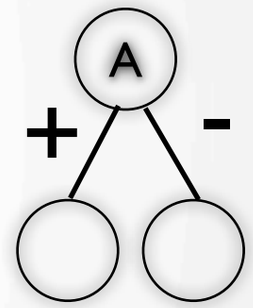
// DEV meetings.pl

Binary space-partitioning tree

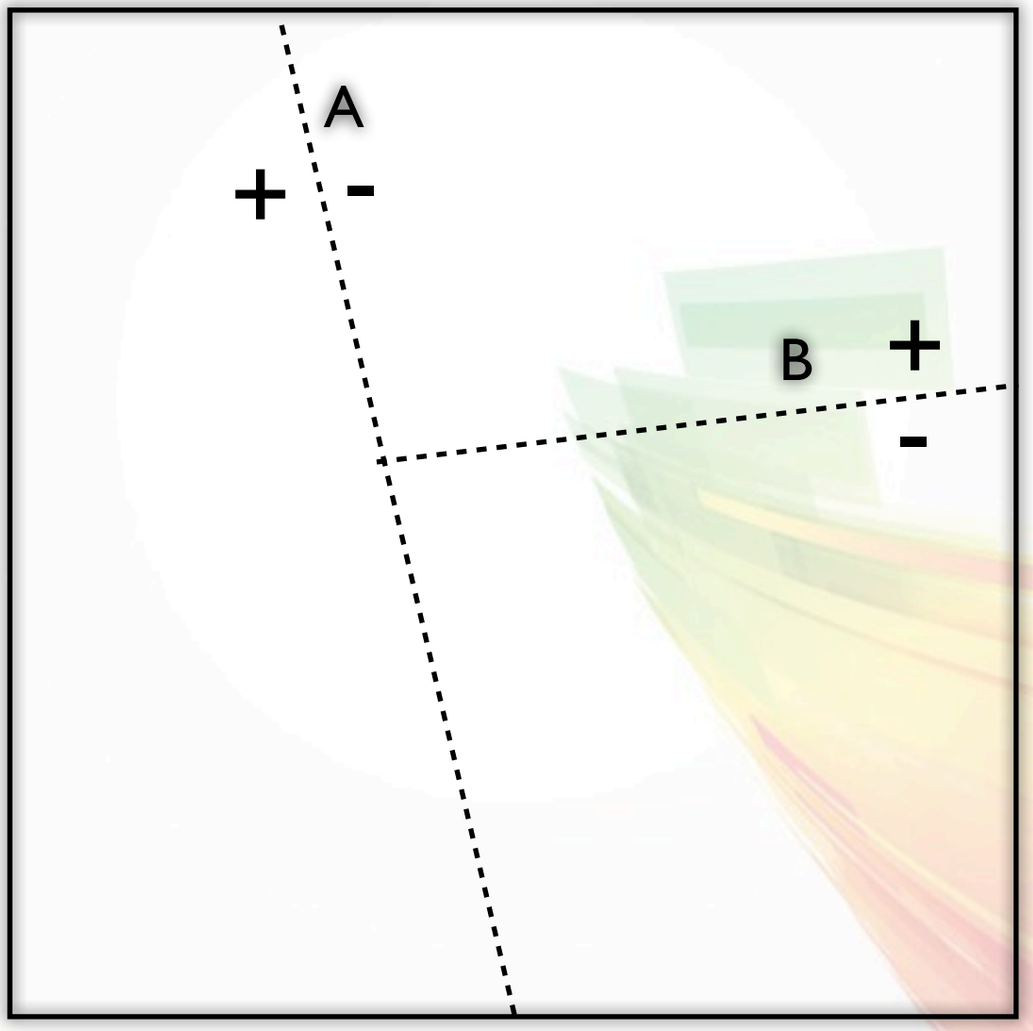
- dividing space using hyperplanes (planes/lines)
- creating positive and negative halfspaces

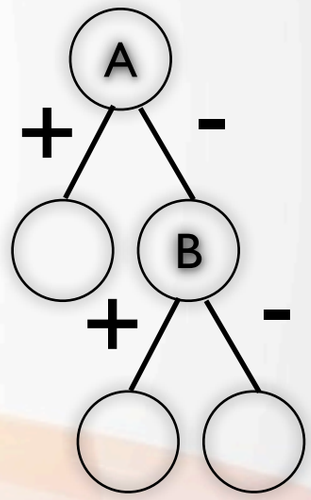
BSP



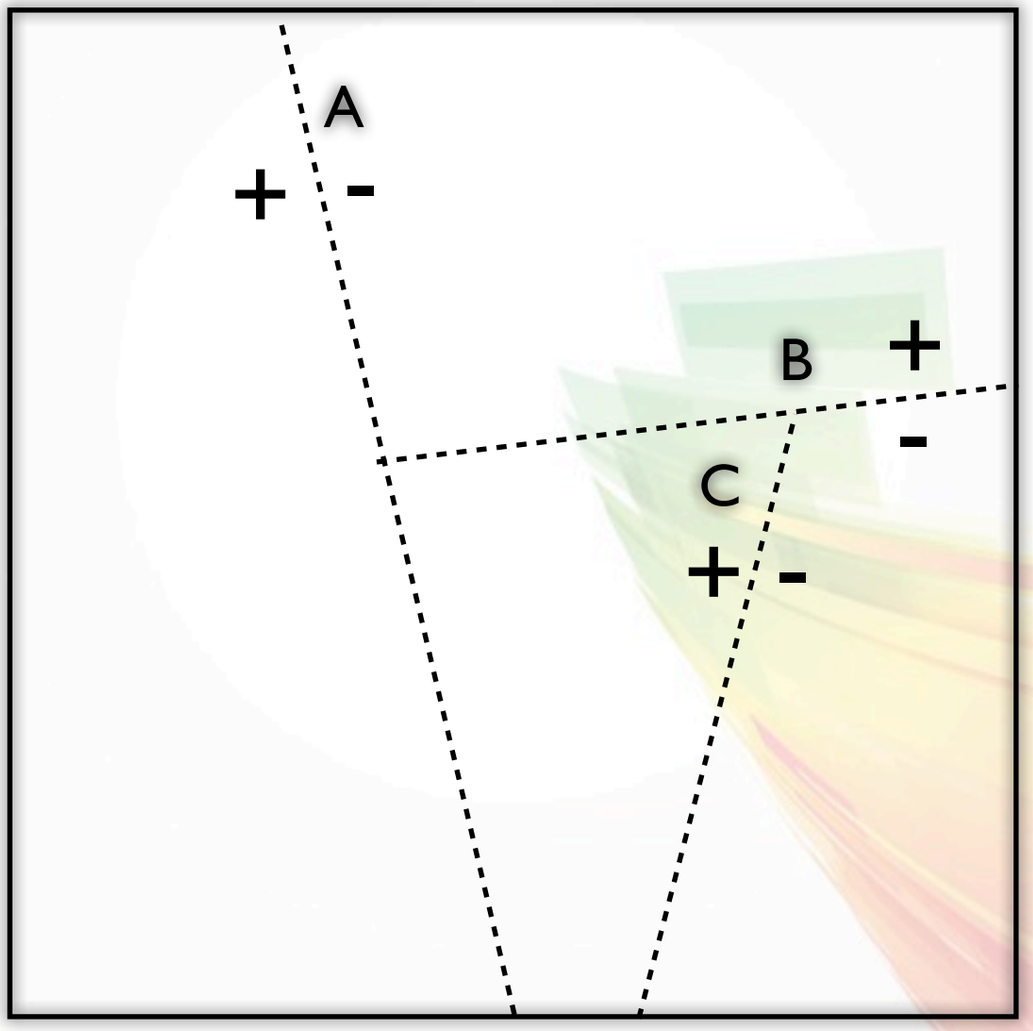


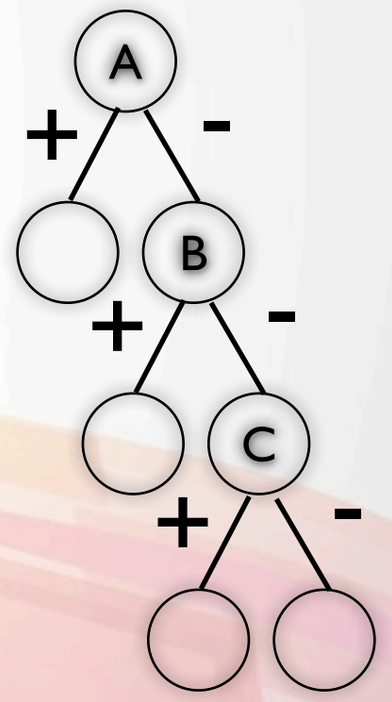
BSP





BSP





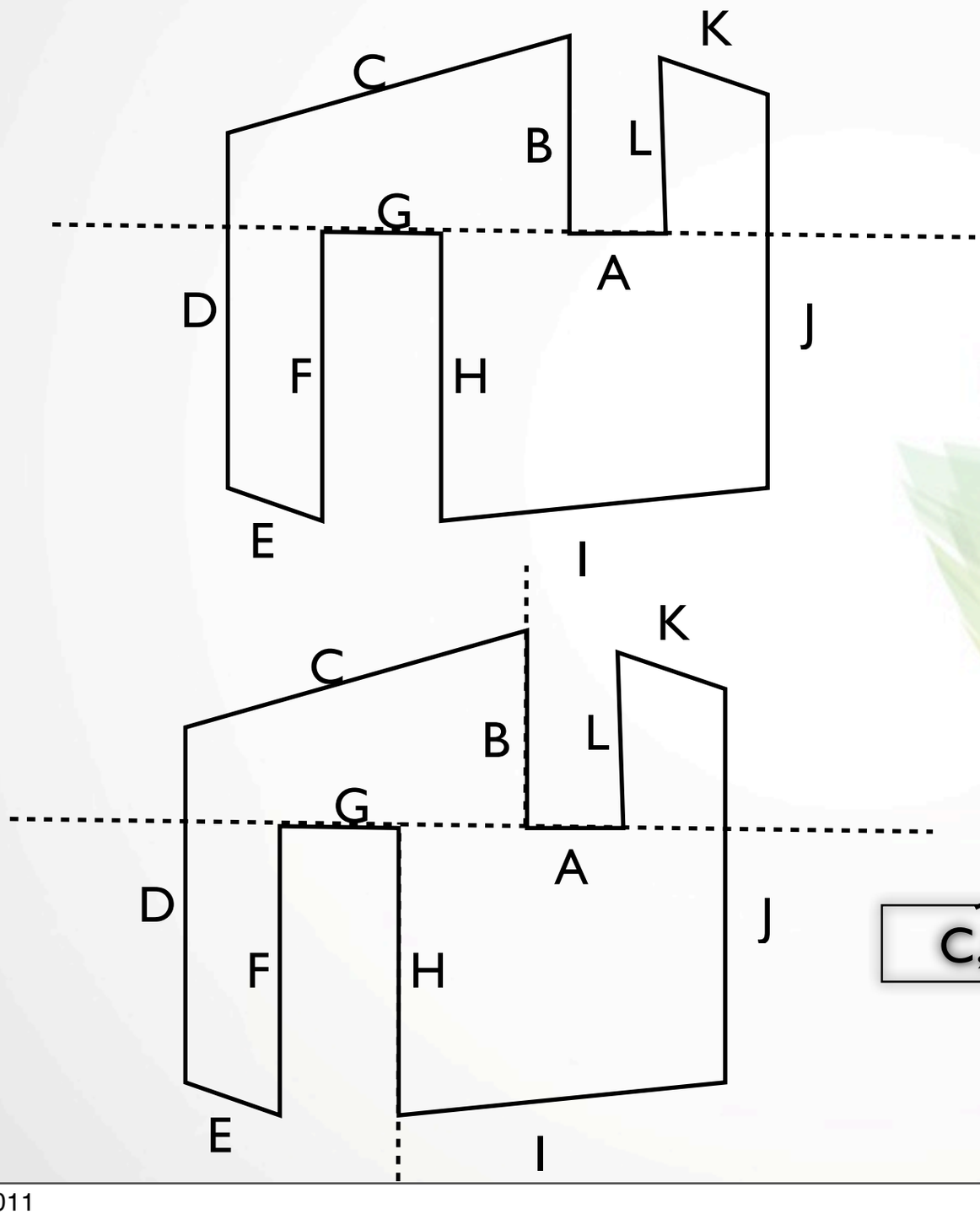
BSP Trees

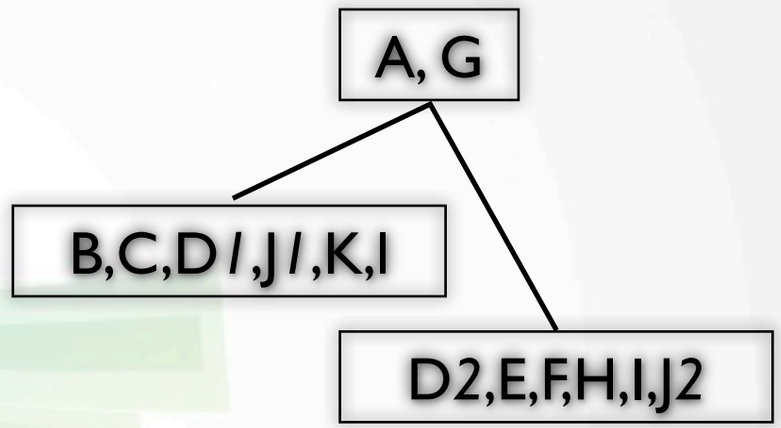
- Node-storing
- Leaf-storing
- Solid-leaf

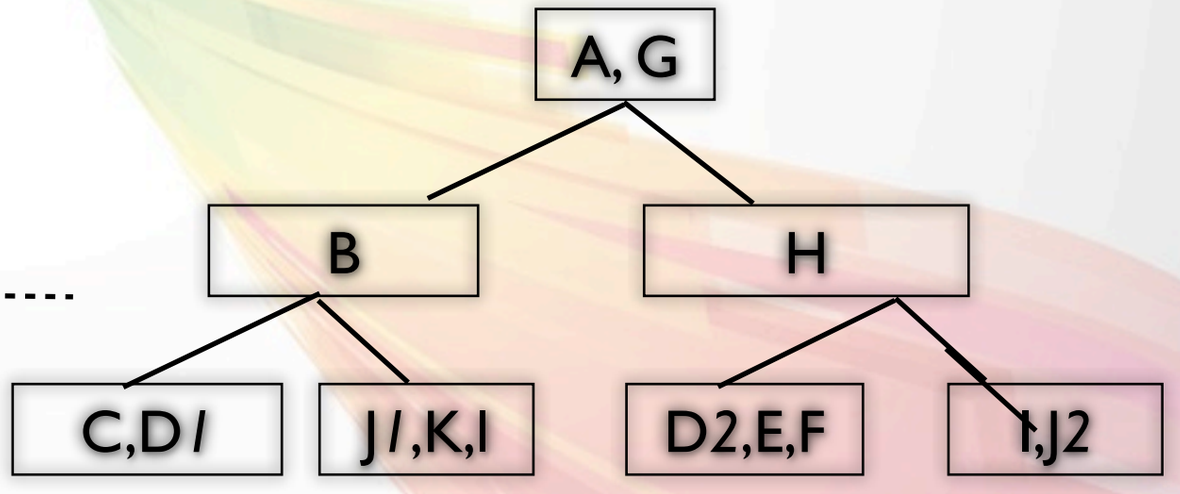
Node-storing

- Auto partitioning
- Traditionally used for rendering of 3d env
- Not suitable for collision detection

Node-storing



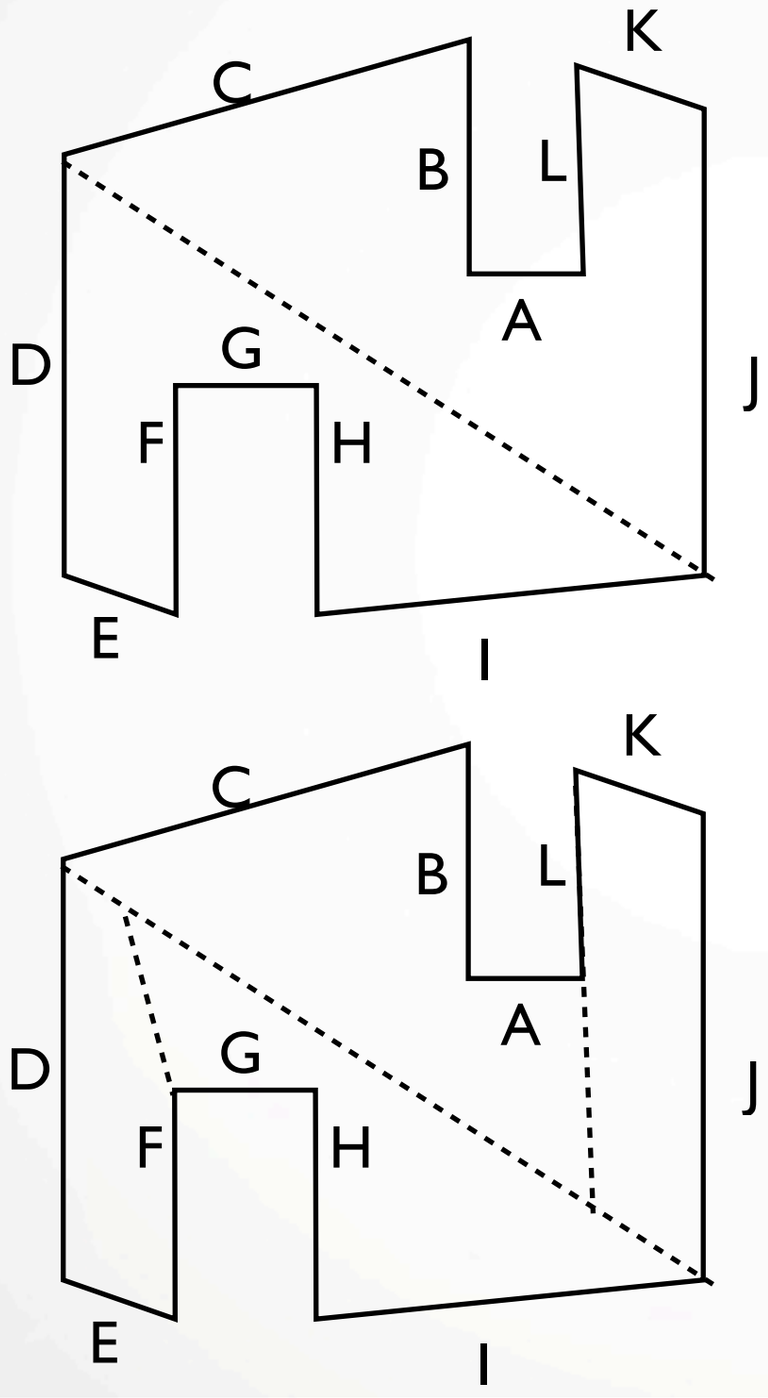


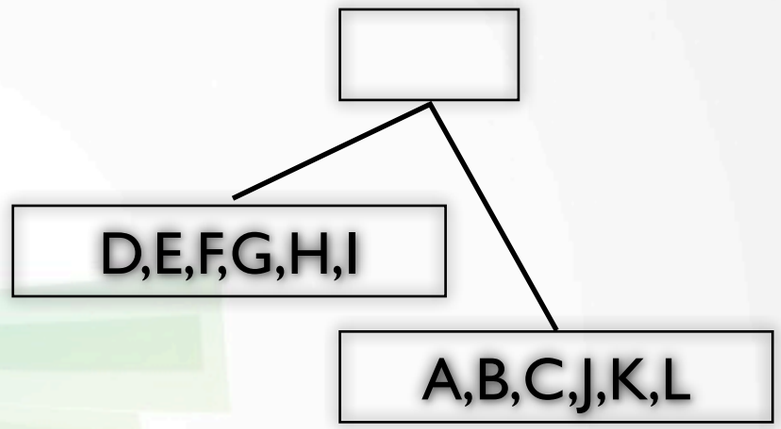


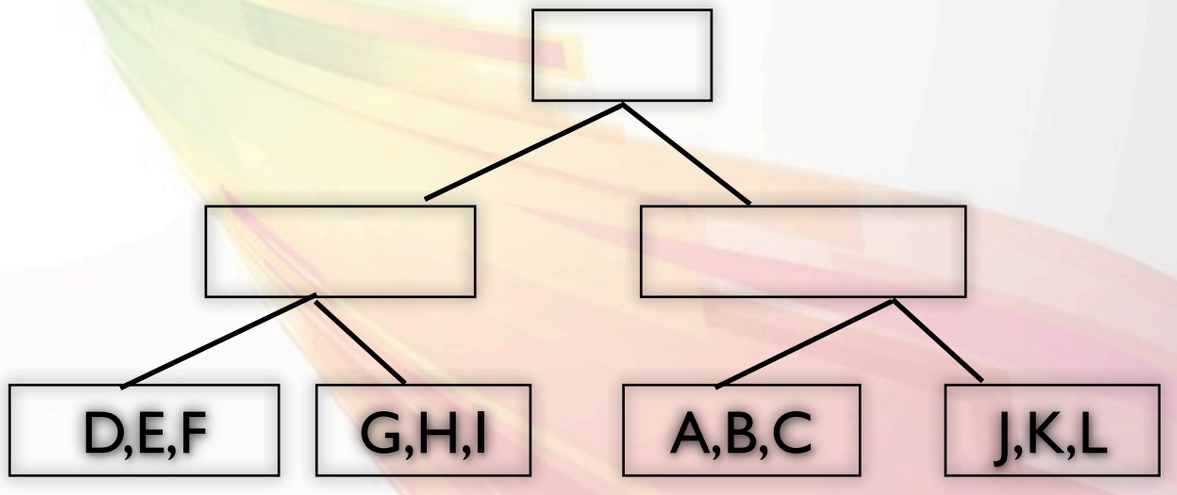
Leaf-storing

- Data is stored rather in the leaves than intrnal nodes
- Internal tree only contains the dividing plane and references to the child subtrees
- Queries do not suffer from having to test unnecessary polygons when traversing
- Collision queries perform tests on al polygons contained in the leaves overlapped by the query

Node-storing



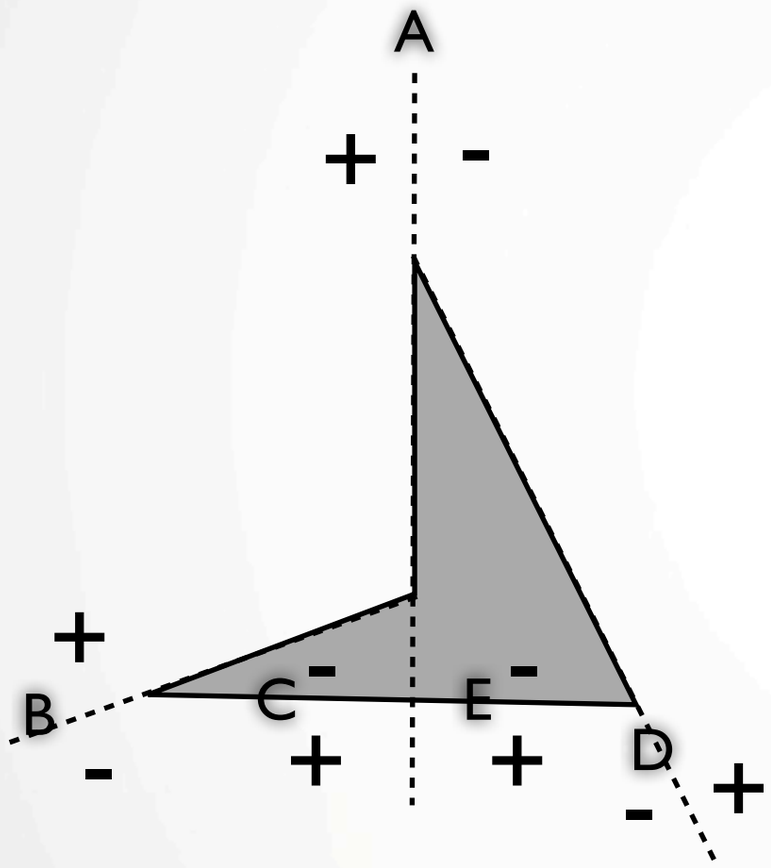


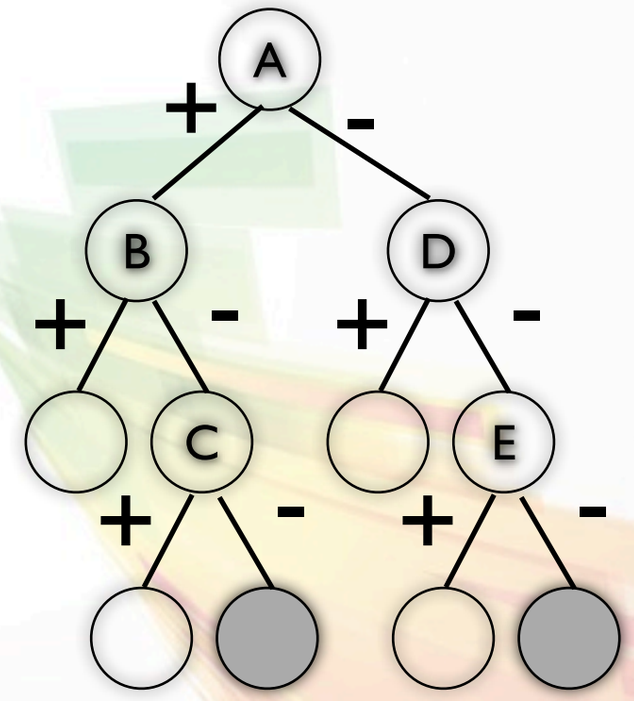


Solid-leaf

- built to represent the solid volume occupied by the input goemetry

Solid-leaf





= empty

= solid

Hybrid

- Rendering world in quake2
  - http://www.flipcode.com/archives/Quake_2_BSP_File_Format.shtml
- Collision detection in Quake3
  - http://www.devmaster.net/articles/quake3collision/

Quad-/Oct-tree

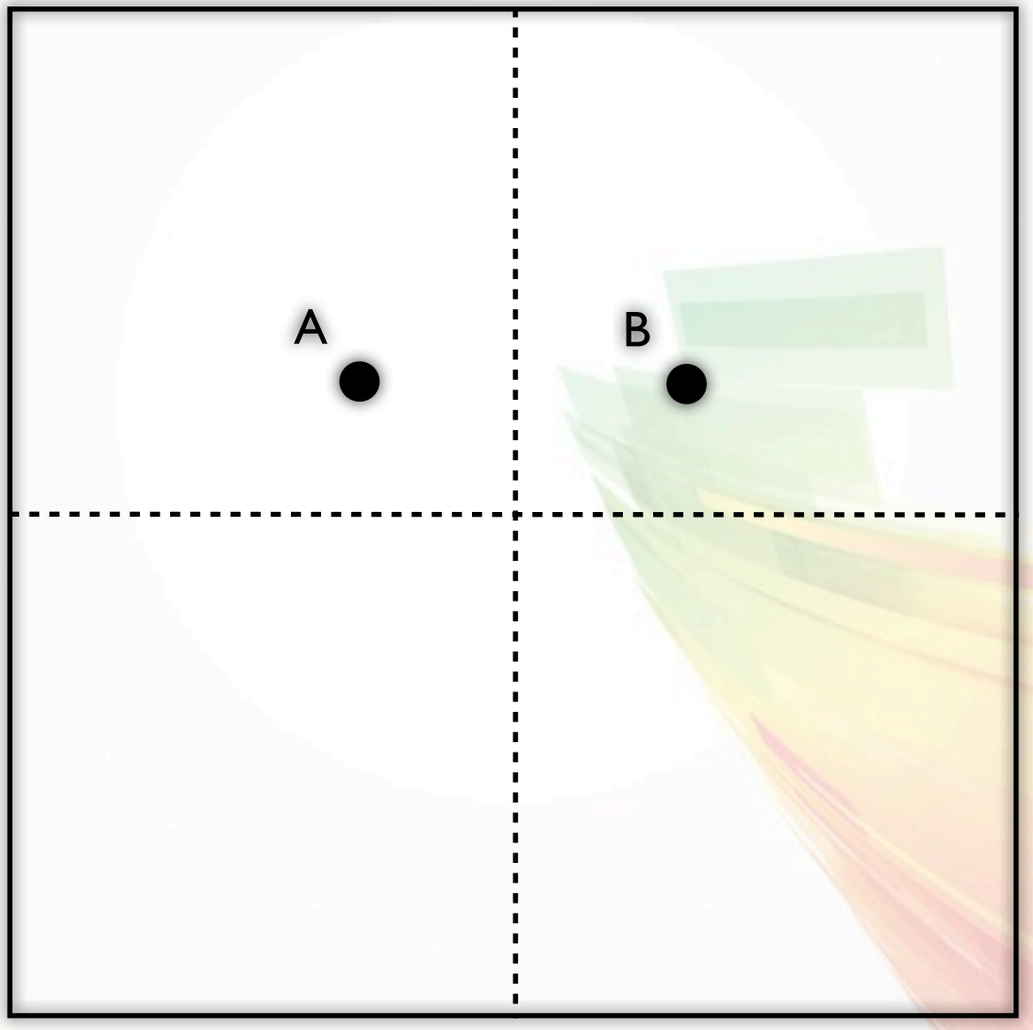
- An axis-aligned hierarchical partitioning method
- Each parent has eight/four children
- Each node has a finite volume associated with it

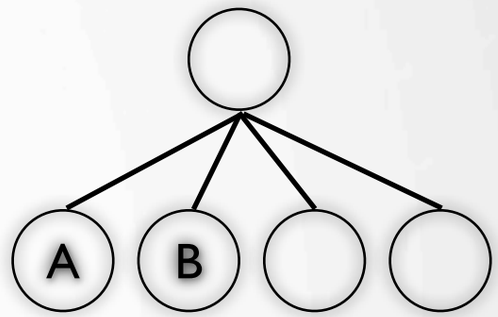
// DEV meetings.pl

Quad-/Oct-tree

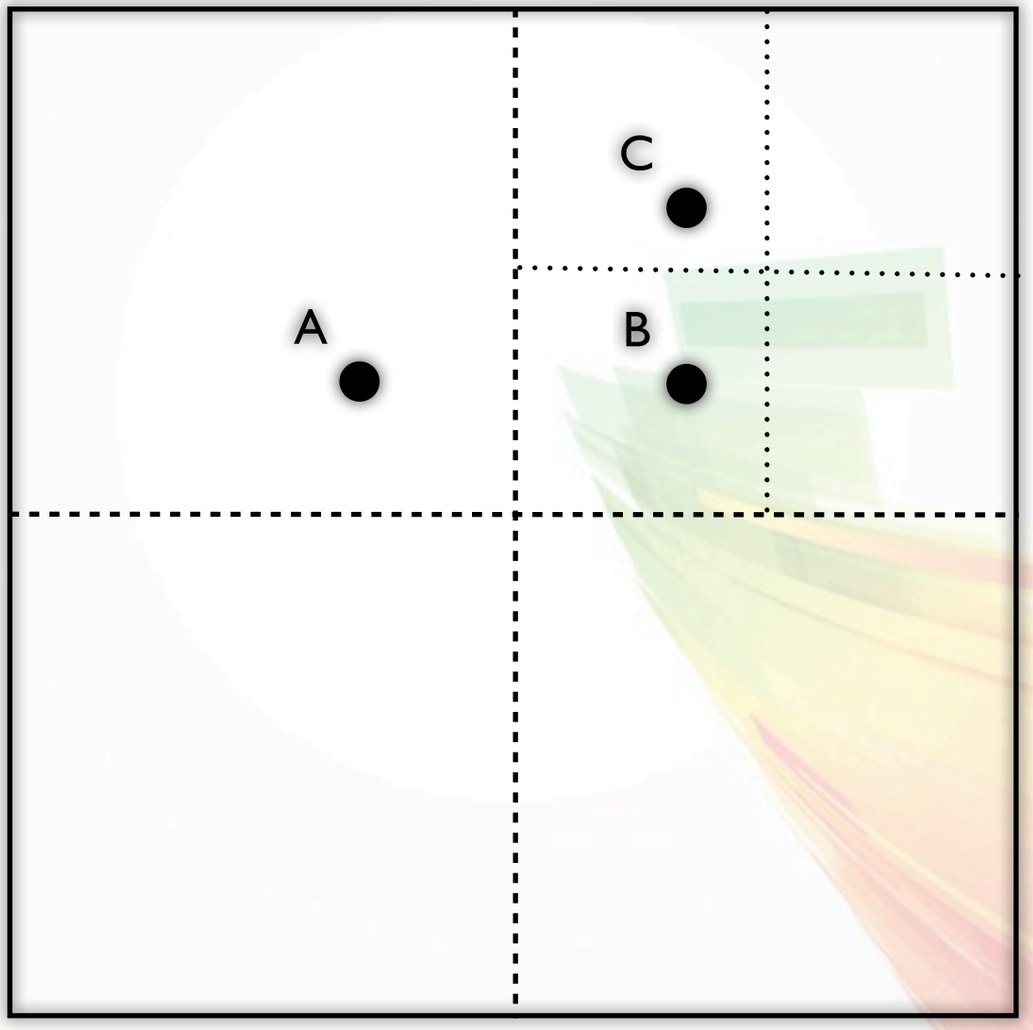
// DEV meetings.pl

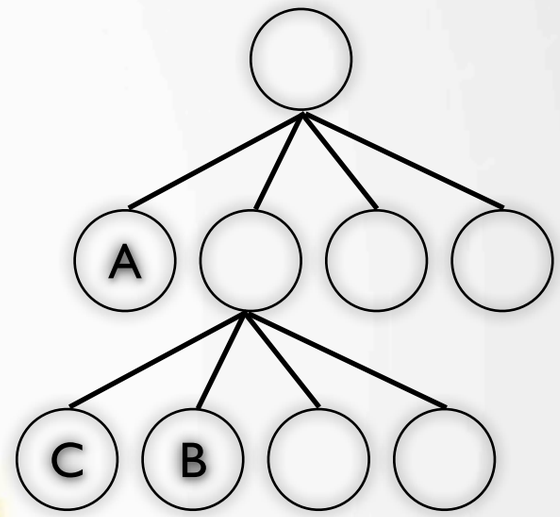
Quad-/Oct-tree



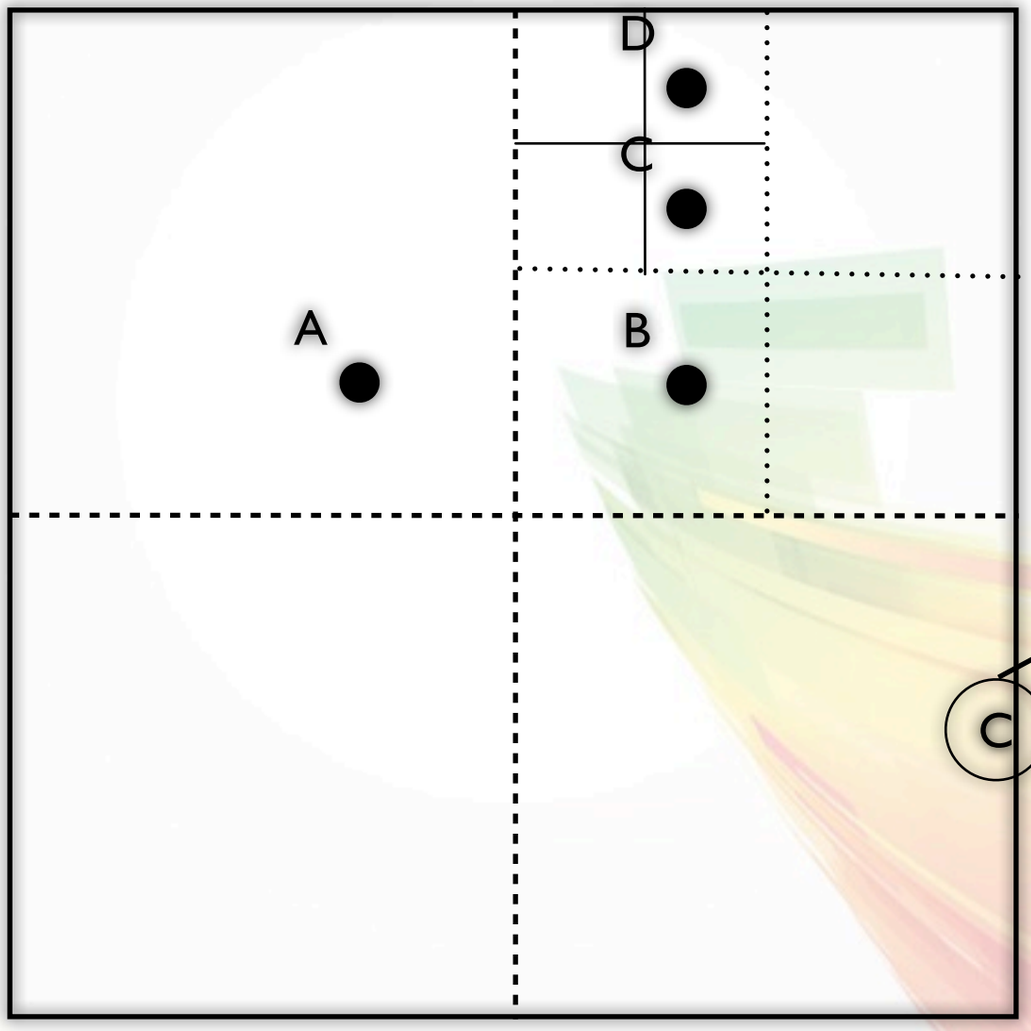


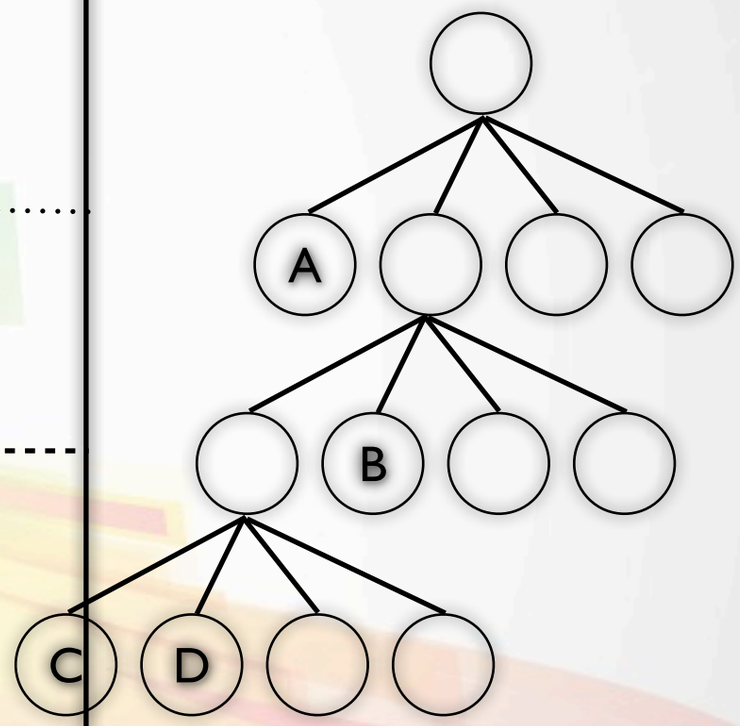
Quad-/Oct-tree



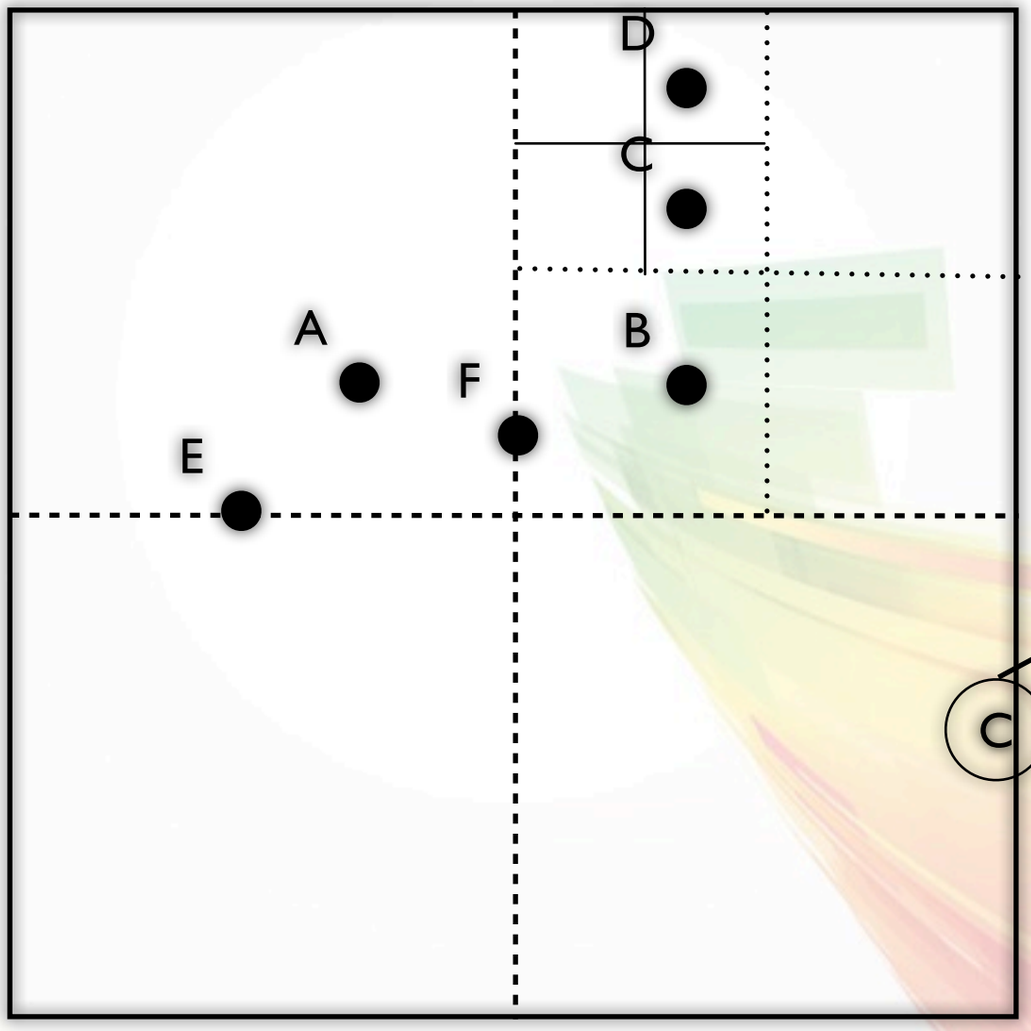


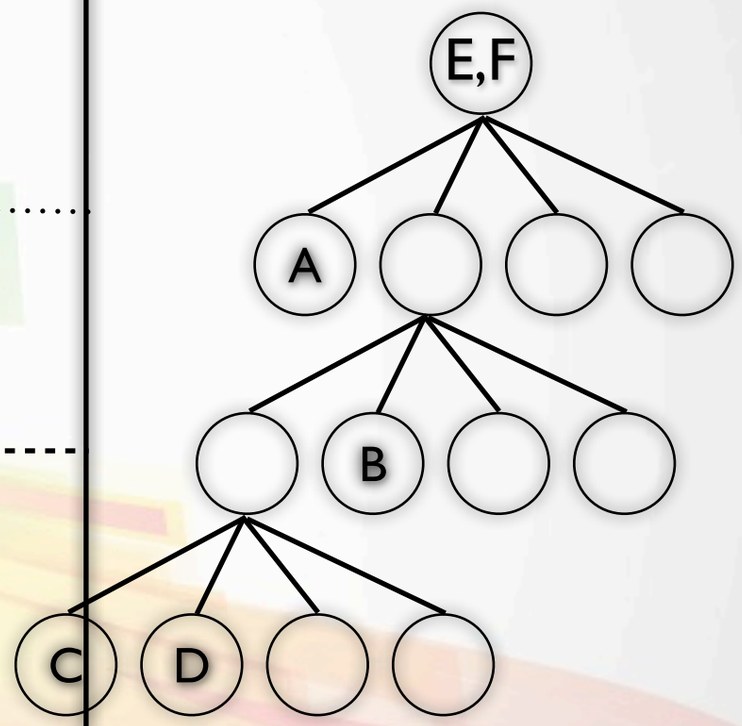
Quad-/Oct-tree



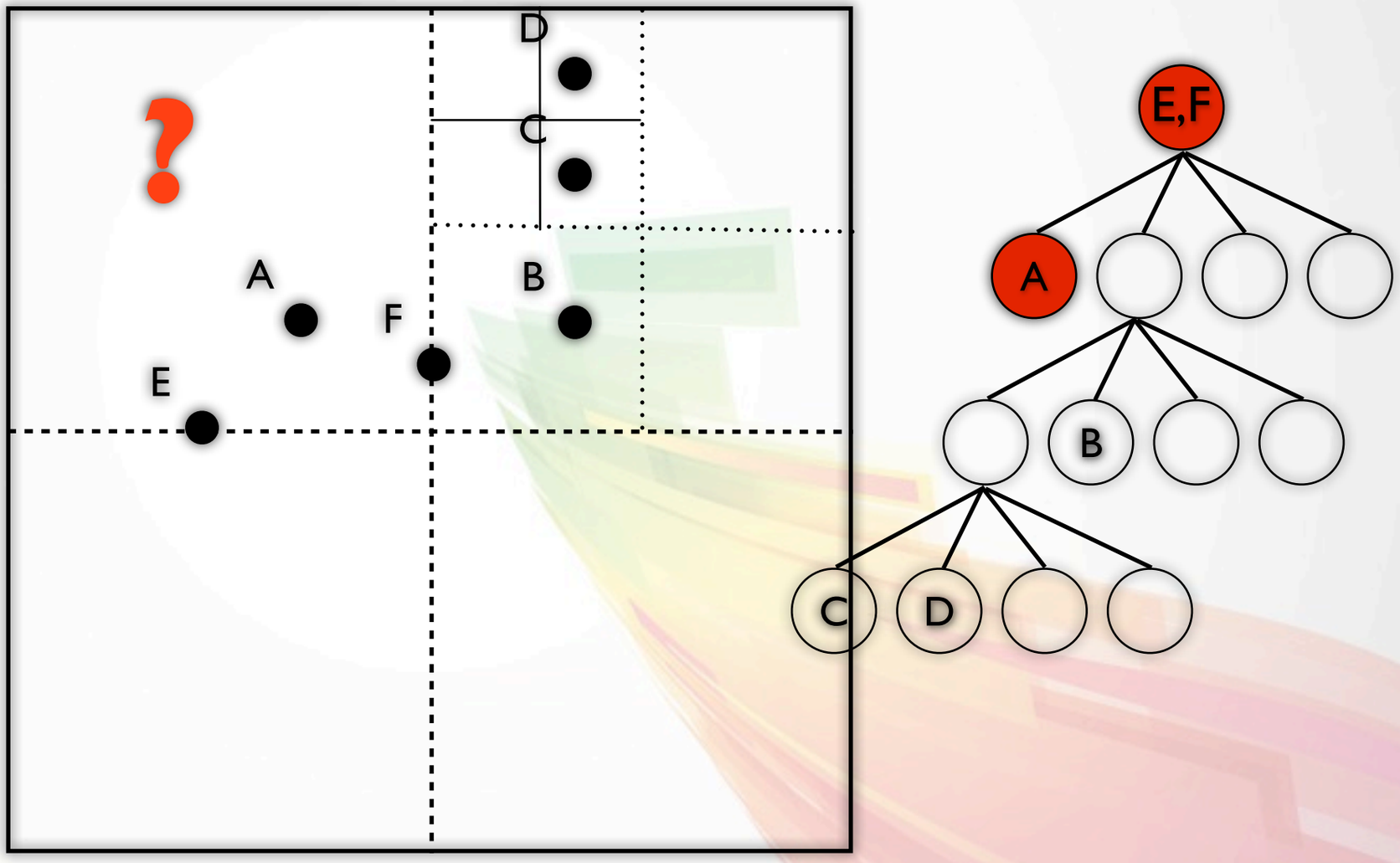


Quad-/Oct-tree

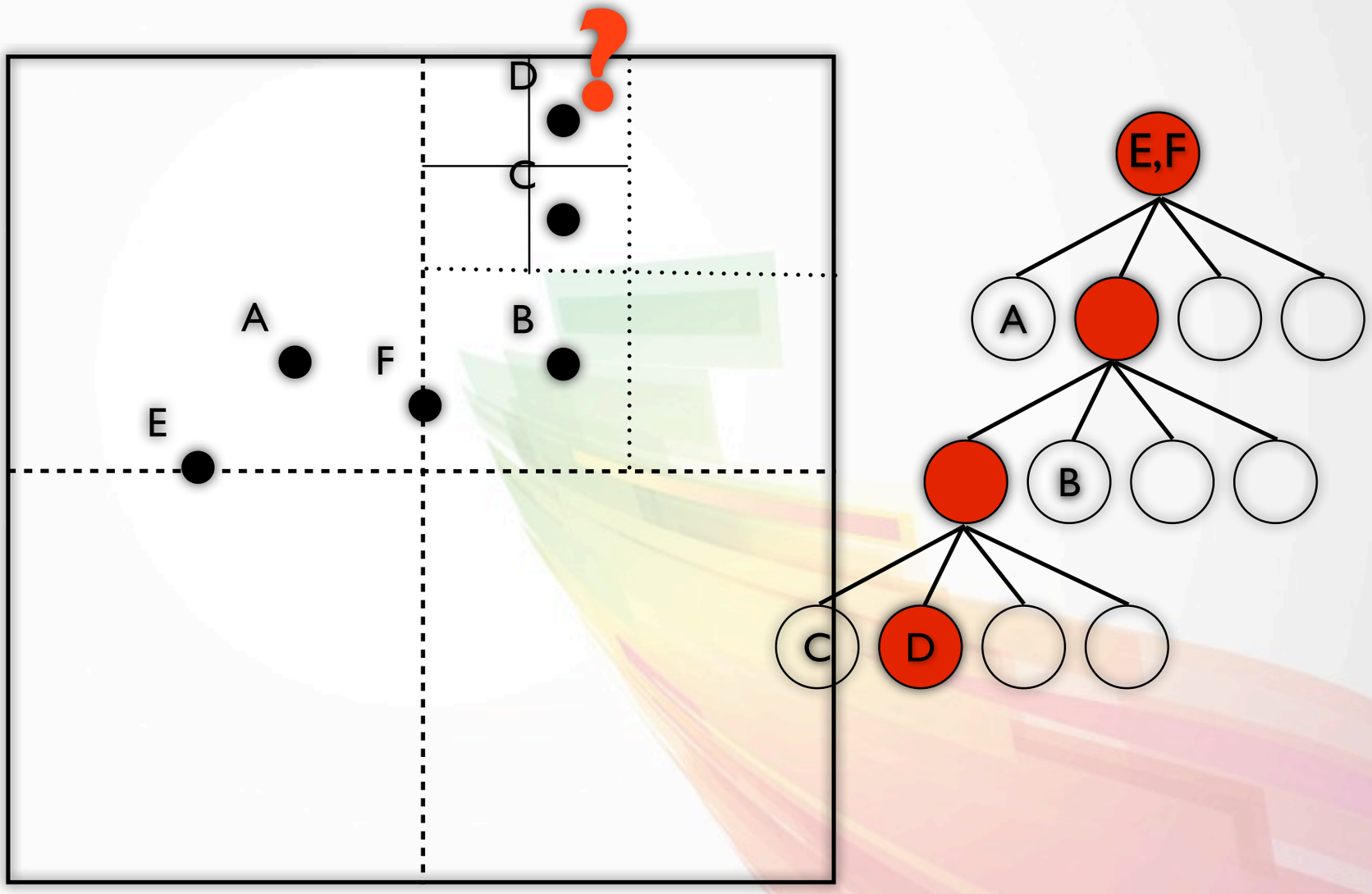




Quad-/Oct-tree



Quad-/Oct-tree



// DEV meetings.pl

Quadtree in js

- http://www.mikechambers.com/blog/2011/03/21/javascript-quadtree-implementation/

Task

- Add broad phase checking using quad tree implementation

// DEV meetings.pl

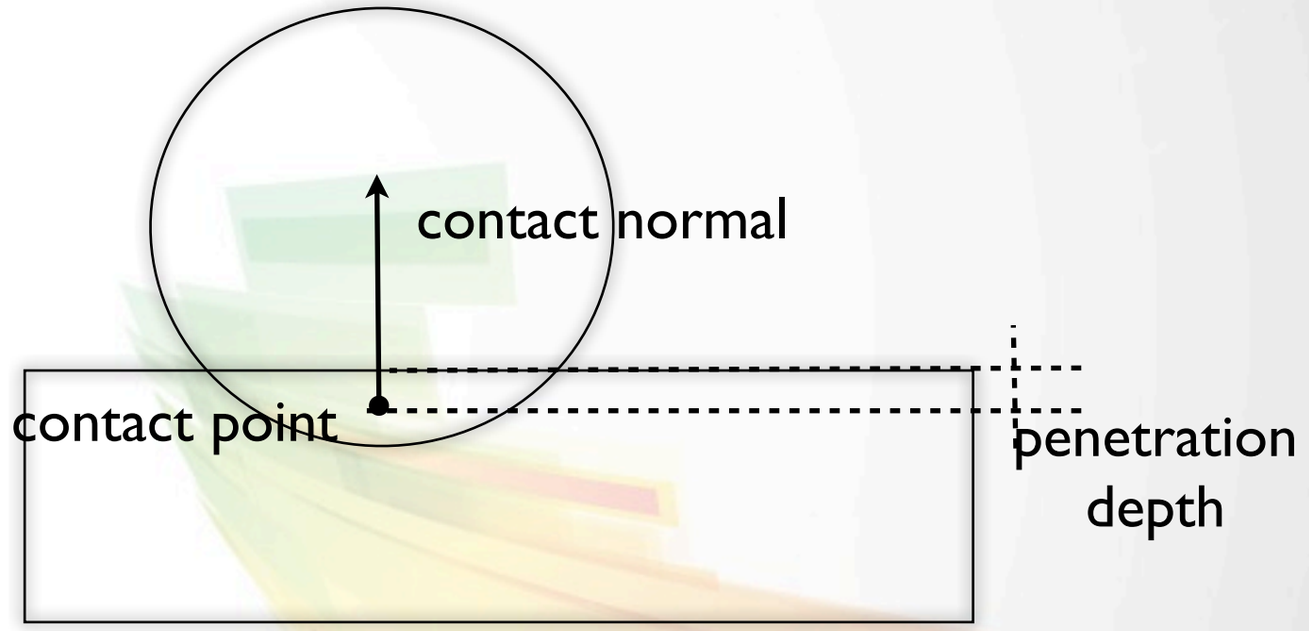
Reaction on collision

Contact type

- Vertex-face
- Edge-Edge
- Edge-Face
- Face-Face

Contact data

- collision point
- collision normal
- penetration depth
- restitution/bounce
- friction/slide



Contact normal and separating velocity

- $n^ = (p_a - p_b)^$, where ^ is normalization
- $v_s = (v_a - v_b) \cdot n^$
- $v'_s = -cv_s$, where c is scalar quantity called coefficient of restitution

Resolving contact

- Resolve velocity
- Resolve interpenetration

Resolve velocity

- Calculate separating velocity
- Calculate new velocity by multiplying with restitution
- Calculate total inverse mass
- Calculate impulse and impulse per mass
- Apply impulses to objects

Resolve velocity

```
function calculateSeparatingVelocity (a, b) {
    var relativeVelocity = vec3.subtract(a.getVelocity(),
        b.getVelocity(),vec3.create());
    return vec3.dot(
        relativeVelocity,
        vec3.normalize(vec3.subtract(
            a.getPosition(),
            b.getPosition(),
            vec3.create())));
}
```

Resolve velocity

```
function resolveVelocity (a, b) {
    var separatingVelocity = calculateSeparatingVelocity(a, b);

    if (separatingVelocity < 0 ) return;

    var newSepVelocity = -separatingVelocity * restitution;
    var deltaVelocity = newSepVelocity - separatingVelocity;

    var totalInversMass = a.getInverseMass() + b.getInverseMass(); // watch out on infinite masses

    var impulse = deltaVelocity / totalInverseMass;
    var contactNormal = vec3.normalize(vec3.subtract(a.getPosition(), b.getPosition(),vec3.create()));

    var impulsePerMass = vec3.scale(contactNormal, impulse);

    vec3.add(a.getVelocity(), vec3.scale(impulsePerMass, a.getInverseMass(), vec3.create()));
    vec3.add(b.getVelocity(), vec3.scale(impulsePerMass, -b.getInverseMass(), vec3.create()));
}
```

Resolve interpenetration

- movePerMass = vec3.scale(contactNormal, (penetration / totalInverseMass);

- vec3.add(a.getPosition() , vec3.scale(movePerMass, a.getInverseMass(), vec3.create());

- vec3.add(b.getPosition() , vec3.scale(movePerMass, -b.getInverseMass(), vec3.create());

Destruction

- Replacing given colliding object with set of new objects

Reflexing after world planes

- Negation of proper velocity component
- In more realistic simulation add friction and bounce depending on kind of object which collide

Task

- Implement simple collision response for
- Asteroid collide with asteroid
- Asteroid vs bullet
- Asteroid vs world plane

// DEV meetings.pl

Thank You!!!

marekapawlowski@gmail.com
@m4r00p
https://github.com/m4r00