

Mechanics, Dynamics & Aesthetics

- The Complexity of Games
- Expectations of Players
- Efficiency
- Game Mathematics
 - Collision Detection & Response
 - Object Overlap Testing
 - Minkowski Sum
 - Geometry, Trigonometry
 - Vectors
- Game State
- Narrative
 - In literature
 - In film
 - In games
- MDA
 - Mechanics
 - Dynamics
 - Aesthetics

Video Games Are Very Complex Programs

Programming a simple web-hosted Flash game requires a very high level of programming knowledge; may also require expertise in:

- Network Protocols
- Data retrieval and management techniques
- Graphic & Sound creation/management
- Geometry, Trigonometry and Calculus
- Vector mathematics
- 3D mathematics

Player Expectations

In general, games are held to a higher standard than other types programs.

People expect "office applications" to fail, and don't expect 100% up-time from business websites.

What we are willing to tolerate when "working" is wildly different then what we are willing to tolerate when "playing".

Efficiency

Complexity and computability are concepts that are not normally taught on an undergraduate level.

Game programmers need to consider "efficiency" in everything they do.

If a player had to wait more than 30 seconds for levels to load in a game, their "review" of that game was greatly reduced.

Game Mathematics

"Game Mathematics" refers both to areas of general mathematics (geometry, trigonometry, calculus) as well as specialized areas of mathematics (vectors, matrices).

Graphic libraries, game libraries, 2D and 3D libraries exist for programming languages to help simplify the mathematical problems that you will face. But they can't be relied on to do everything.

Collision Detection

Figuring out if two objects are touching incredibly common problem in a game.

Two basic techniques:

1. Overlap testing

- Detects whether a collision has already occurred

2. Intersection testing

- Predicts whether a collision will occur in the future

Overlap Testing

Facts:

- Most common technique used in games
- Exhibits more error than intersection testing

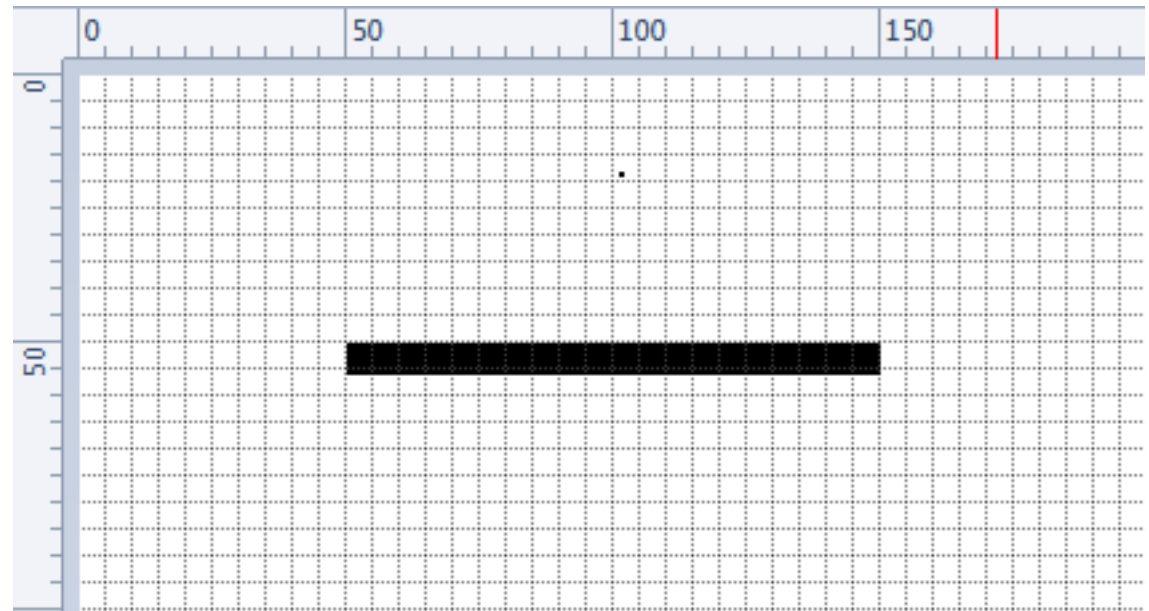
Concept

- For every simulation step, test objects to see if they overlap
- Easy for simple volumes like spheres, harder for polygonal models

Simple Overlap Testing

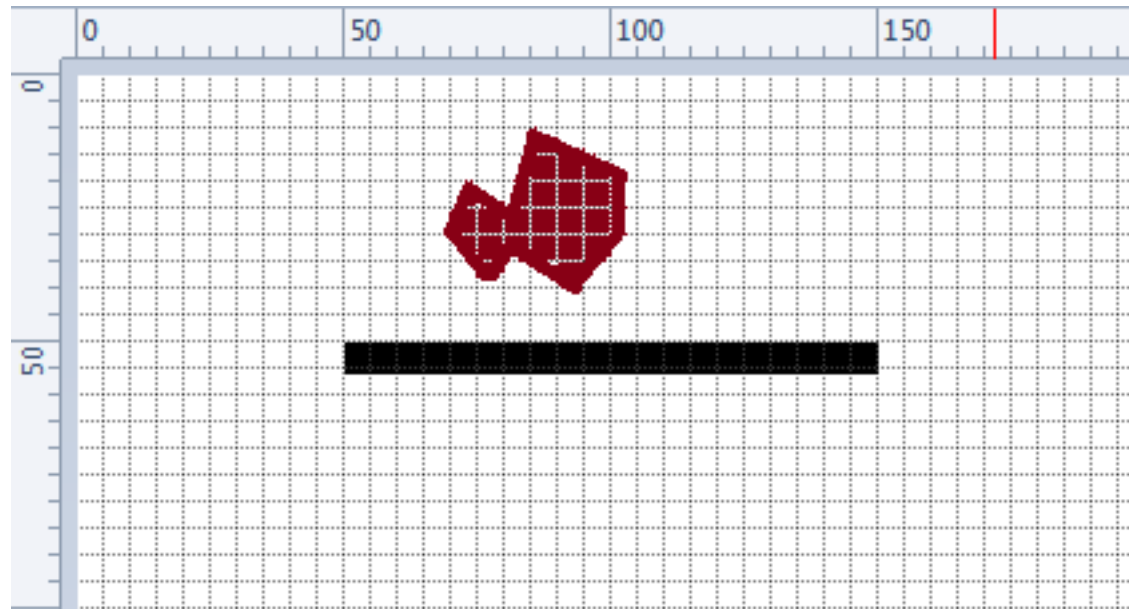
Simple example is particle interacting with a square. This will still require 4 logical tests in a 2D game.

Depending on the type of game that played, the order of those 4 tests can have a profound effect on efficiency.



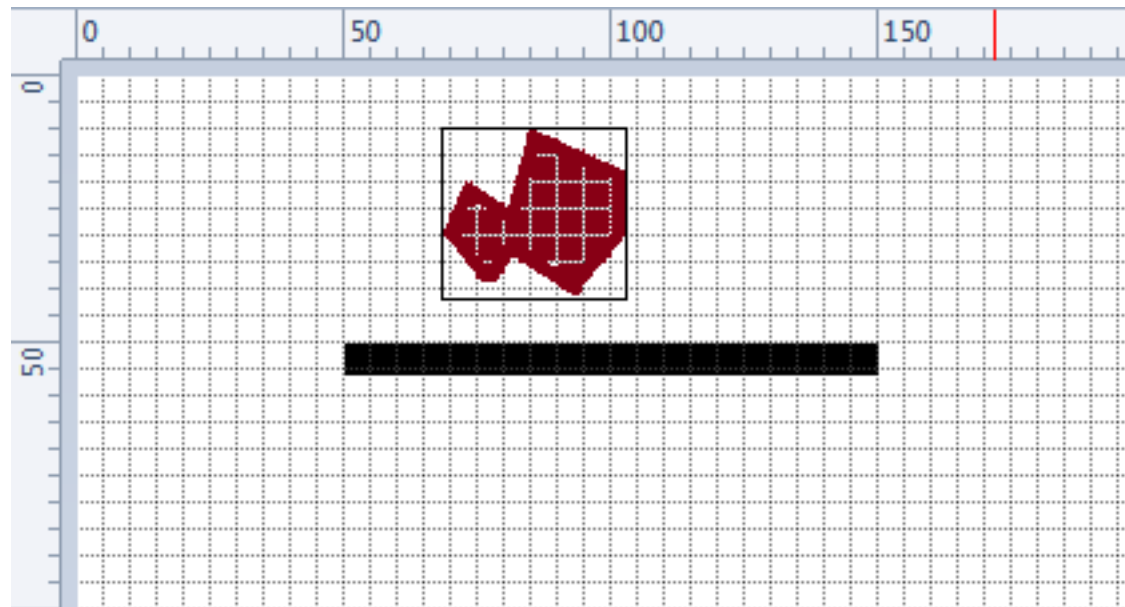
Complex Shape OOT

How many tests would be required now?



Bounding Boxes

Bounding Boxes can be used to reduce the complexity of shapes to simplify overlap testing. Note that secondary testing may need to be done if the bounding box is found to overlap. How many test now?

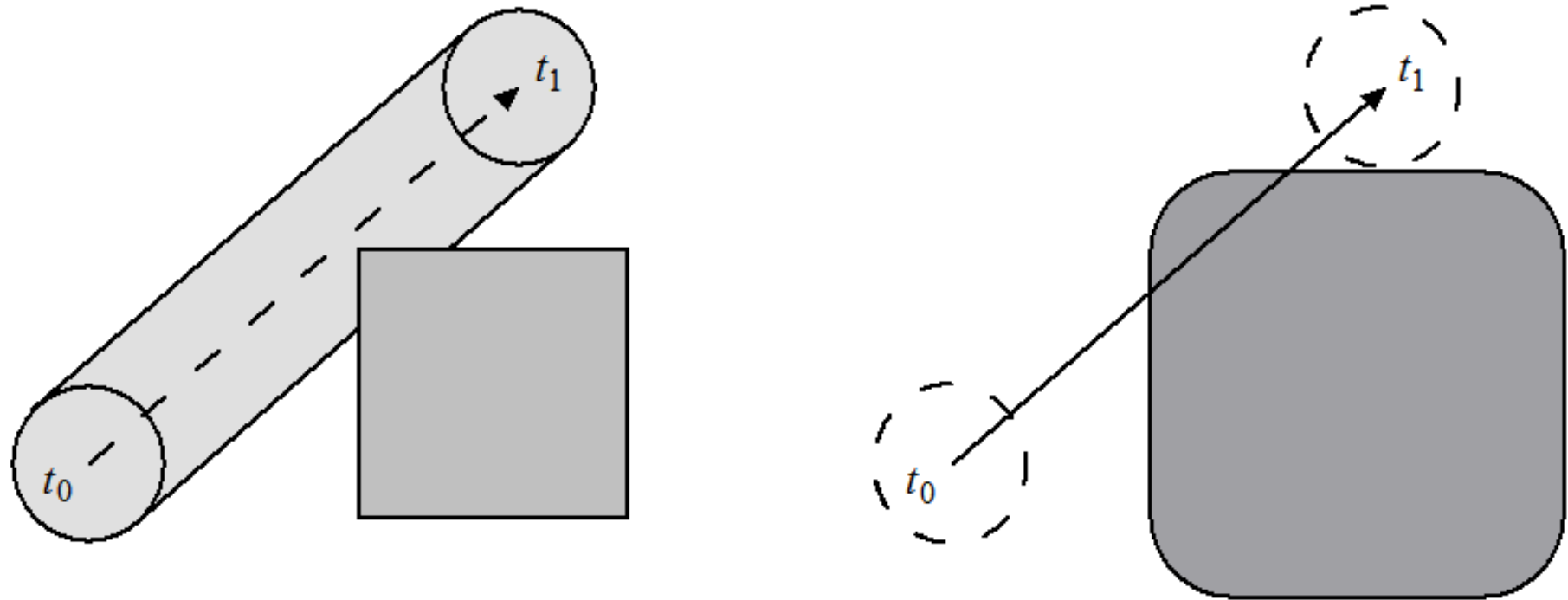
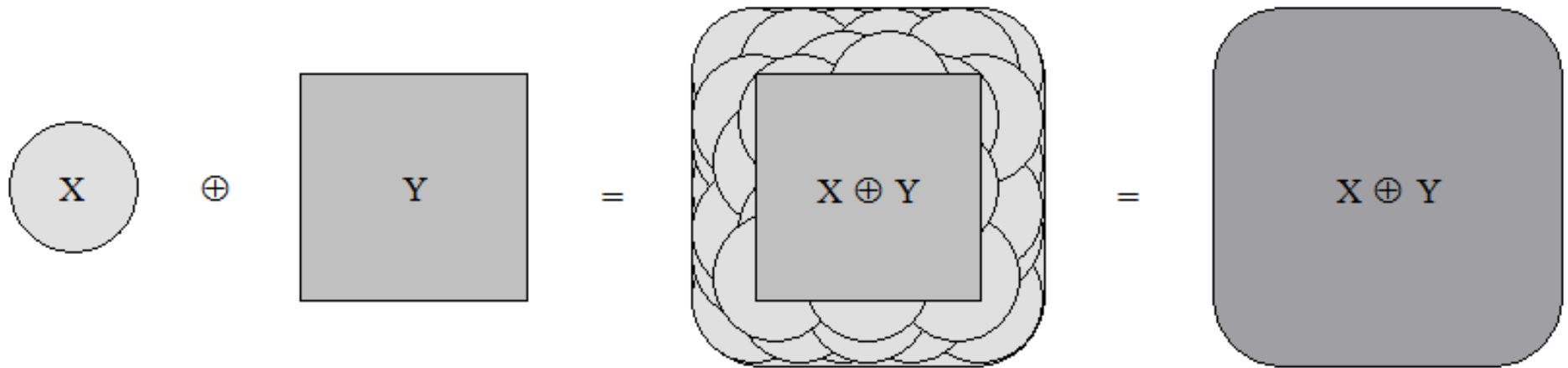


Minkowski Sum

By taking the Minkowski Sum of two complex volumes and creating a new volume, overlap can be found by testing if a single point is within the new volume.

Variations on the Minkowski Sum include calculating the x, y and z distances between the two objects that are being tested.

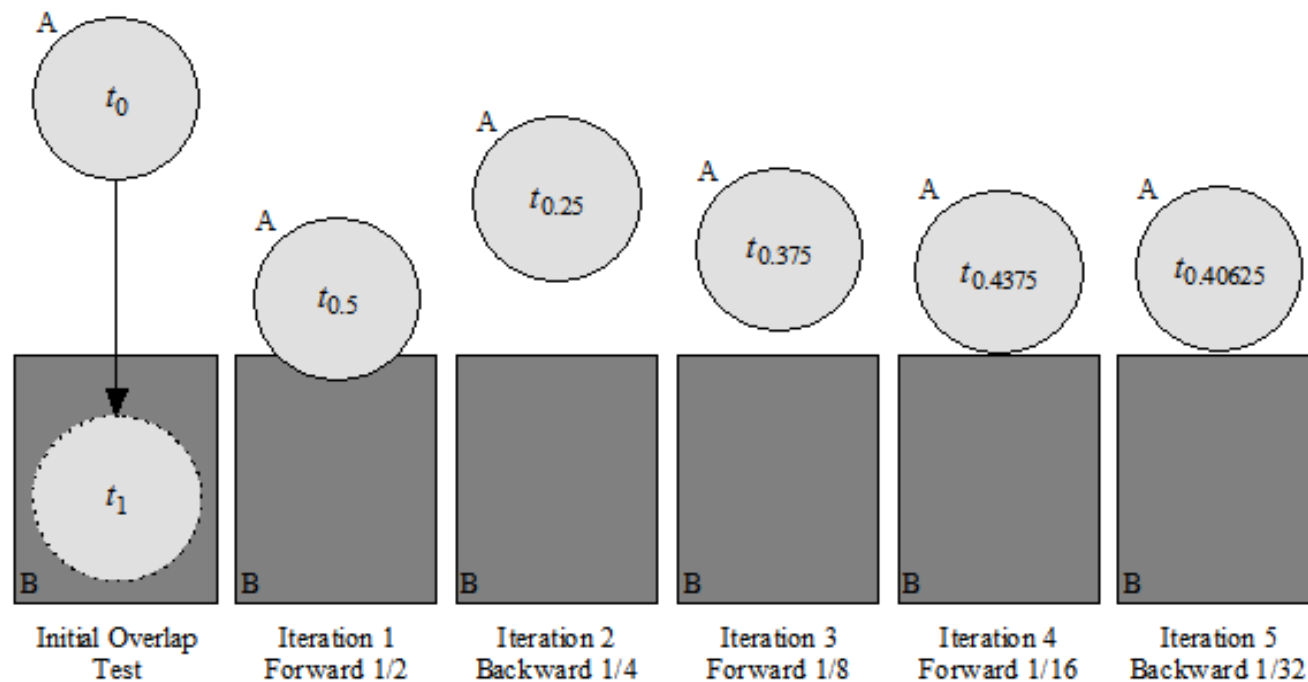
Minkowski Sum



OT - Collision Time

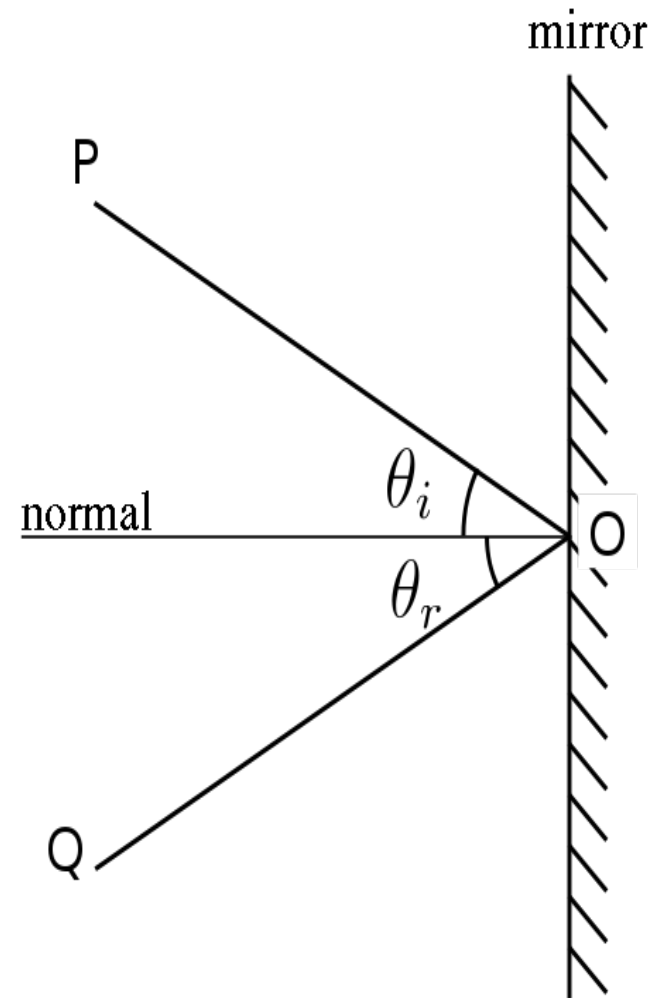
Collision time calculated by moving object back in time until right before collision.

- Bisection is one effective technique.
- Minkowski values are another.



Collision Response

Having captured the exact moment and position of collision, geometry, and trigonometry can be applied to calculate new trajectories.



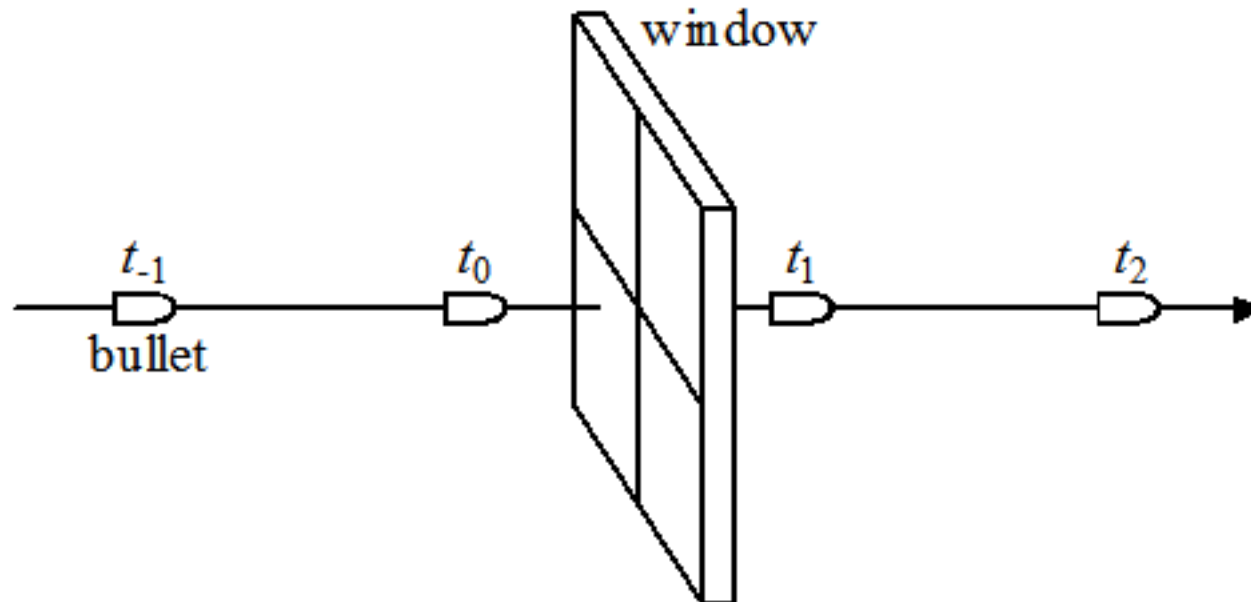
Limits of OT

OT is easy, but limited.

- Fails with objects that move too fast
- Unlikely to catch time slice during overlap

Possible solutions

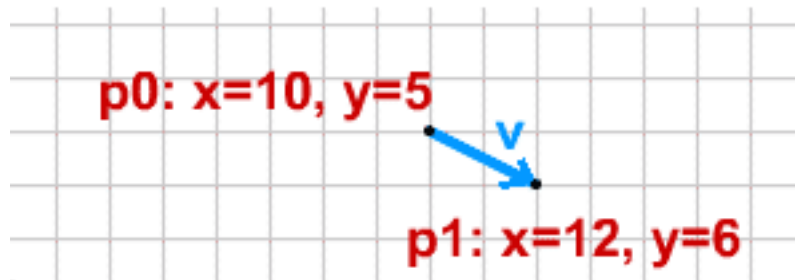
- Design constraint on speed of objects
- Reduce simulation step size
- Use Vectors



Vectors

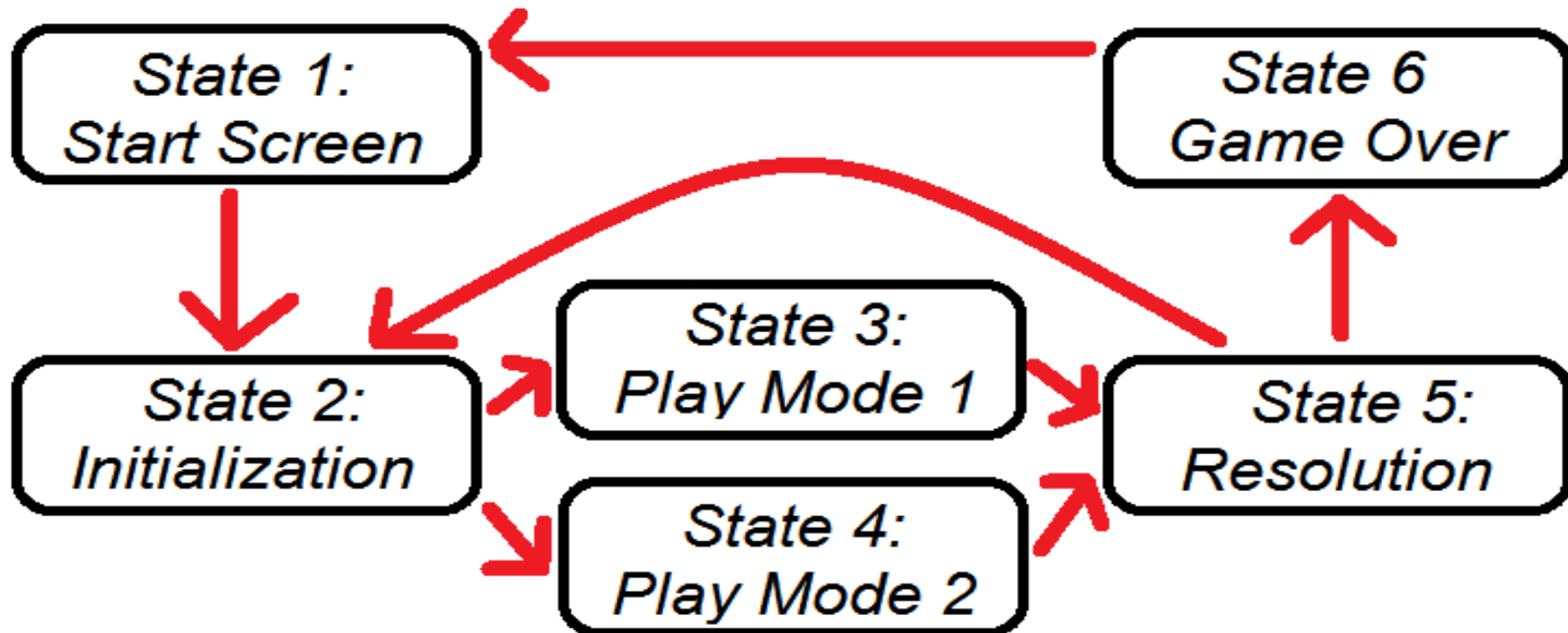
- You already know of "vector images", images represented by a mathematical formula.
- We can represent entire objects (and their movement) with formula's as well.
- Vector (an matrix) mathematics can then be applied to reveal information about where objects will be and whether or not they will collide (at any time, past or future).

`P1[ x1, y1, x2, y2];` `// A particle vector.`



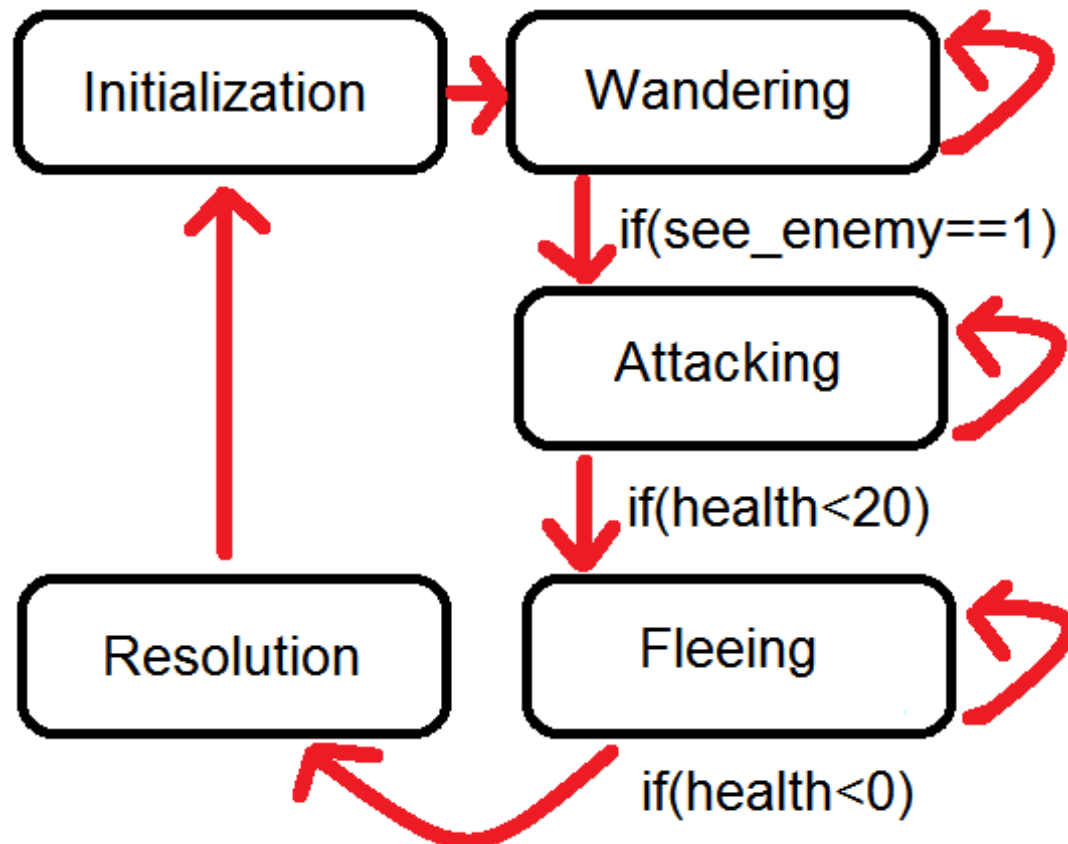
Game State

All games consist of a sequence of states.
Each state is characterized by a combination of visual, audio and/or animation effects, as well as a set of rules that are being applied.



Object State

Objects in the game proceed through their own states as well. These states are defined by the behavior and functionality applied at that time.



Narrative

The narratological view of game studies is that games should be understood as novel forms of storytelling and can thus be studied using theories of narrative.

The "novel form" they refer to is that the player may have some choice about the states that the narrative reaches and/or the order in which the states proceed ("Choose your own adventure").

Treating a game as a narrative (or including sound narrative as part of a game) can help us make a more compelling game.

Narrative in Literature

Rules for narrative in literature have been around since the time of the Greeks (Aristotle's Poetics).

Questions to ask:

1. Whose telling the story?
2. What is the conflict?
3. Who is the player meant to identify?
4. What do you want the player to feel?

Narrative in Film

Modern games have far more in common with film (cinematography) than with regular literature. Cinema also has a lexicon of well established rules regarding the creation of compelling narrative.

1. Don't break the narrative plane.
 2. Don't break the narrative chain.
 3. Use the camera to frame action.
 4. Use the camera to immerse the viewer.
- (Note: You have perfect camera, light, etc.)

Narrative in Games

Ultimate goal (as with literature, and cinema) is to get the player or viewer to "suspend disbelief" and have a "real" emotional response to events that are entirely fictitious.

Including a compelling narrative in a game can "make it incredible" (ChronoTrigger, BioShock) or simply create a series of annoying cut scenes that a player has to wade through.

MDA

Mechanics, Dynamics & Aesthetics

MDA is a game development paradigm designed to help developers make the most out of a game idea, and proceed efficiently through the complex process of bringing a game to market.

MDA is one of many development paradigms that are rigidly used by large game development companies.

Mechanics

Before a single line of code is written the mechanics that will be used by the game should be well thought out and documented.

This includes:

- The programming language
- The programming libraries, engines, tools
- The hardware required/available
- The logical programming components
- The storage/retrieval/initialization methods

Dynamics

Before a single line of code is written the dynamics that will be used by the game should be well thought out and documented. This is the "ludo-logical" part of MDA. All objects and axioms need to be detailed!

This includes:

- The domain of the game.
- The players in the game.
- The rules of the game.
- The objects in the game.

Aesthetics

Before a single line of code is written the aesthetics that will be used by the game should be well thought out and documented. This is the "narratological" part of MDA. The "art bible" which should contain every detail of the "look" of the game will come out of this development area.

This includes:

- Color Pallete
- Physical looks for all players
- Lighting plots, schemes, etc.