Game AI Programming

Procedural Content Generation

Jeremy Gow
j.gow@gold.ac.uk

Procedural content generation is “the **algorithmic creation** of **game content** with limited or indirect user input”

(Togelius et al 2014)

Procedural content generation is “the **algorithmic creation** of **game content** with limited or indirect user input”

(Togelius et al 2014)

"Content"

- Levels
- Textures
- Stories, quests, characters
- Items, weapons, vehicles
- Music
- Game rules
- ...

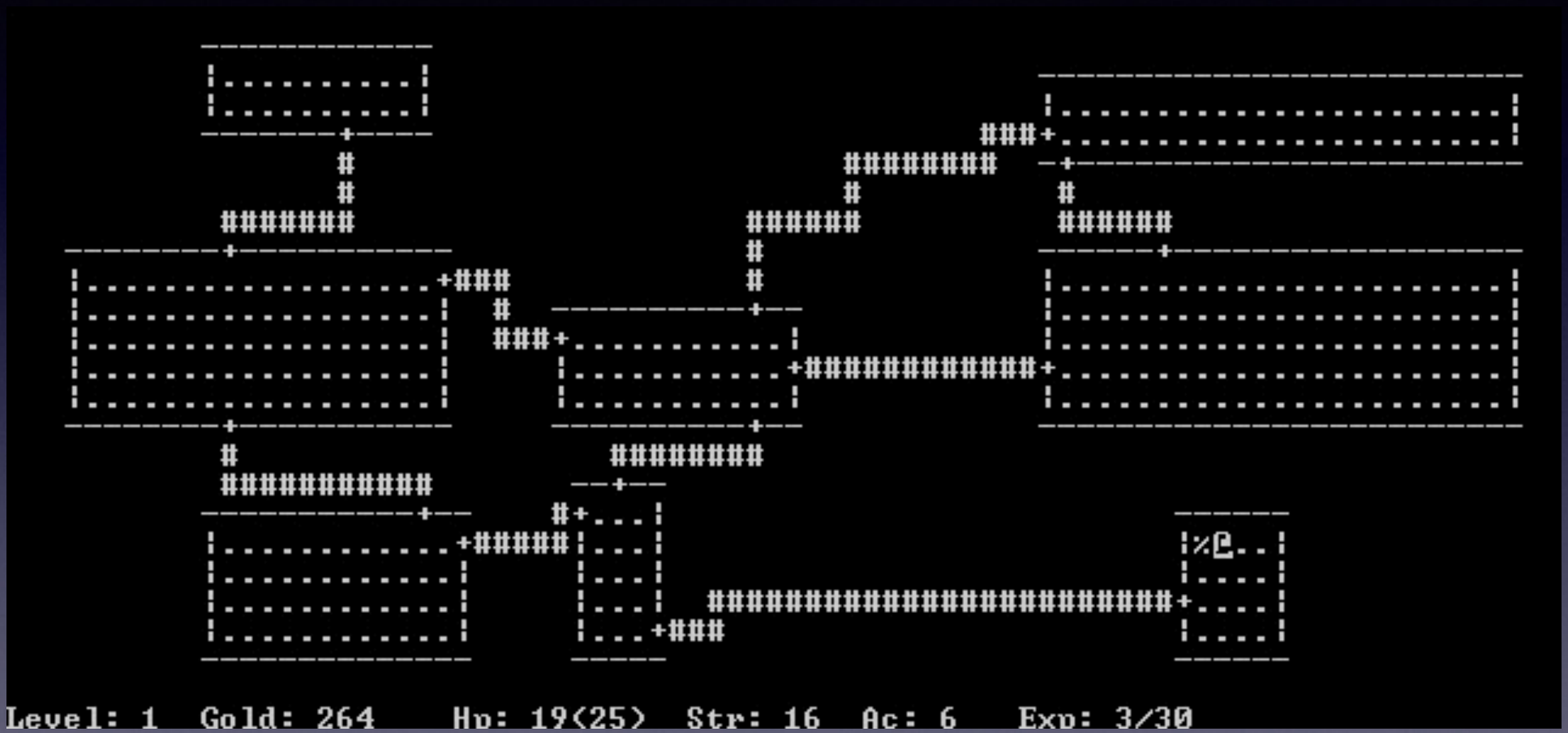
(NPC behaviour not typically described as content)

Games are getting big and expensive

- 1000 people
- 5 years
- >$137 million
- 49 square miles
- 8GB

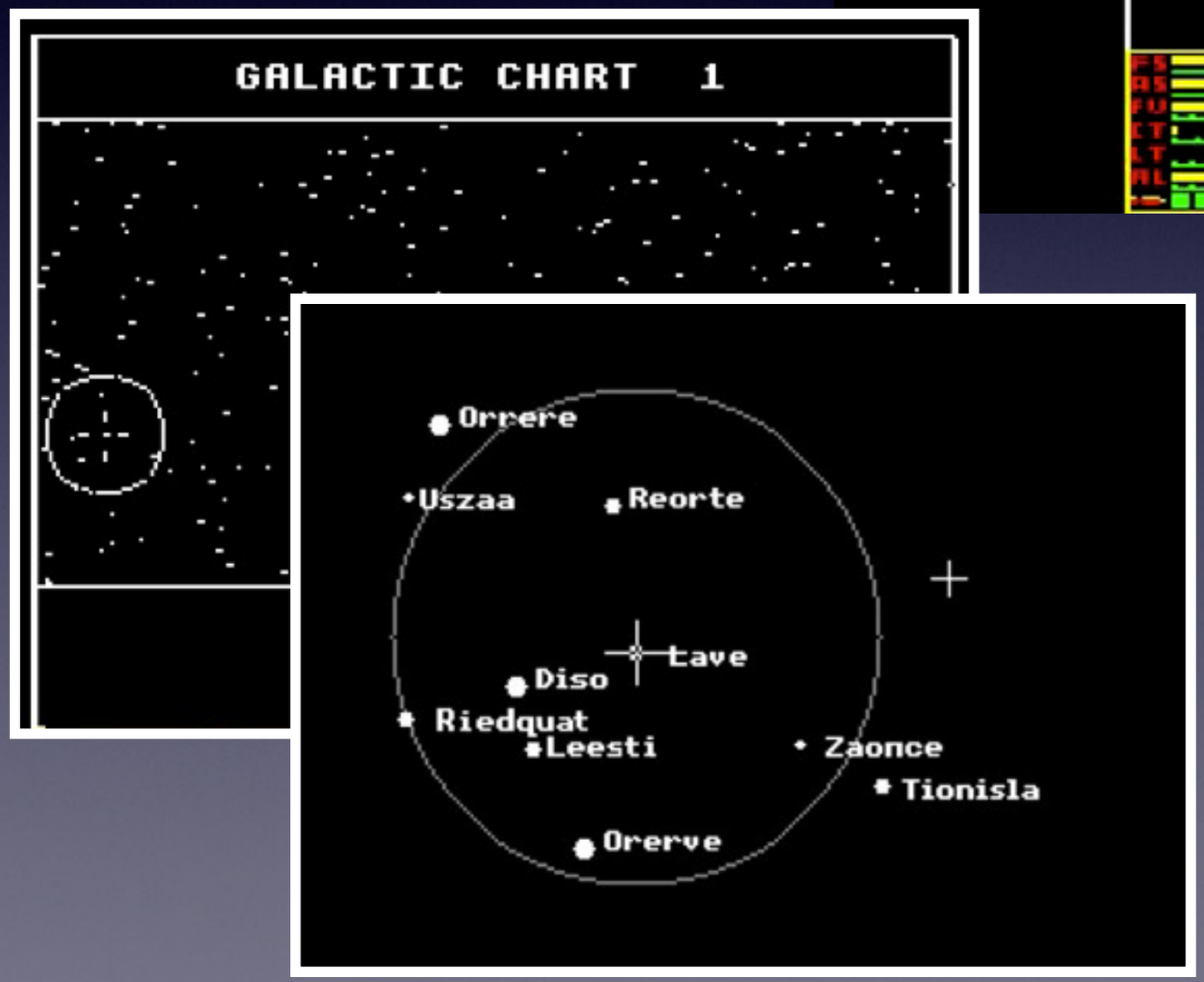
Motivations for PCG

- Economic
    - Cheaper, faster, empowering small teams
- Play
    - Infinite replayable content, personalisation
- Design
    - Inspiration, novelty, understanding



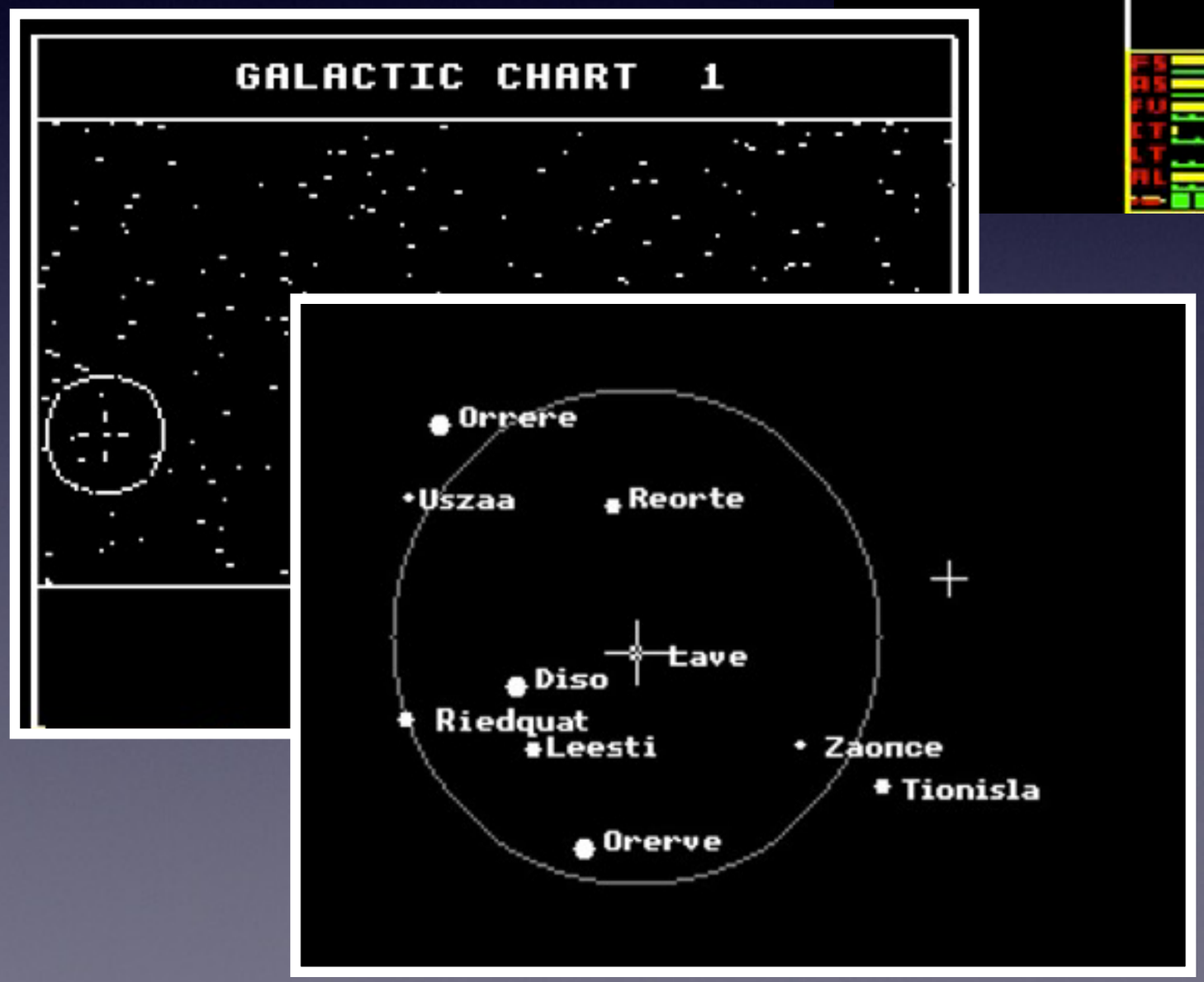
Rogue (1980)

Elite (1984)

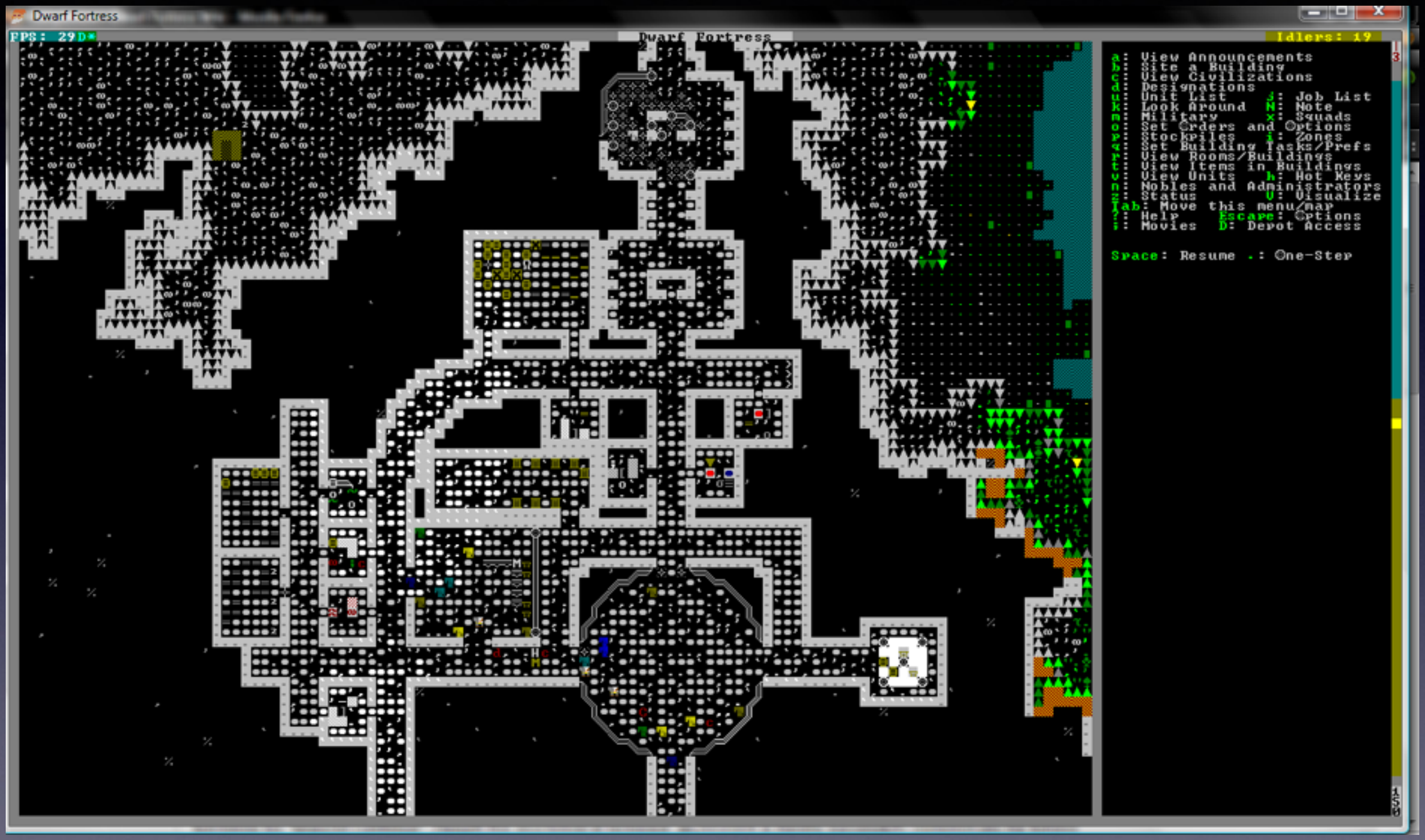


Lave
Radius 4116 km.
Dictatorship, Rich Agri.
Pop. 2.5 B, Prod. 7000 MCr.
HC: 6, TL: 5,
Human Colonials.
Lave is most famous for its vast rain forests and the Lavian tree grub.

Elite (1984)



Lave
Radius 4116 km.
Dictatorship, Rich Agri.
Pop. 2.5 B, Prod. 7000 MCr.
HC: 6, TL: 5,
Human Colonials.
Lave is most famous for its vast rain forests and the Lavian tree grub.



Dwarf Fortress (2006)

Minecraft (2009)

Minecraft (2009)

Diablio III (2012)

Diablio III (2012)

Proteus (2013)

Proteus (2013)

No Man's Sky (2016?)

No Man's Sky (2016?)

PCG Variations

- Online vs offline
- Speed
- Reliability
- Controllability (random seeds, parameters, ...)
- Expressivity and diversity
- Creativity and believability
- Tailored to design of game

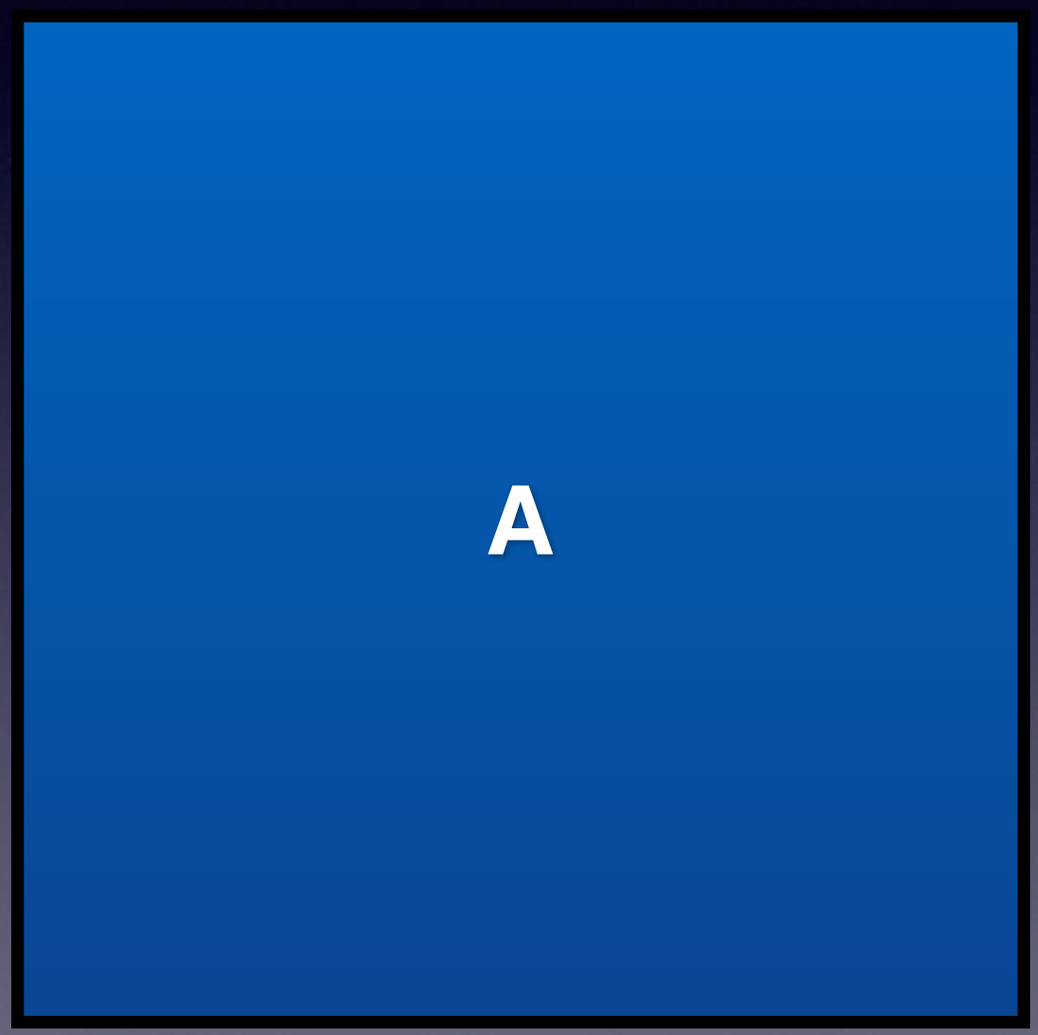
Constructive Generation

- Single-pass algorithms
- Always produce one playable level
- Ad hoc, multistep processes
- Game-specific
- Lacks theoretical foundations (is it A.I.?)
- Fast, run-time generation

Generative Tools

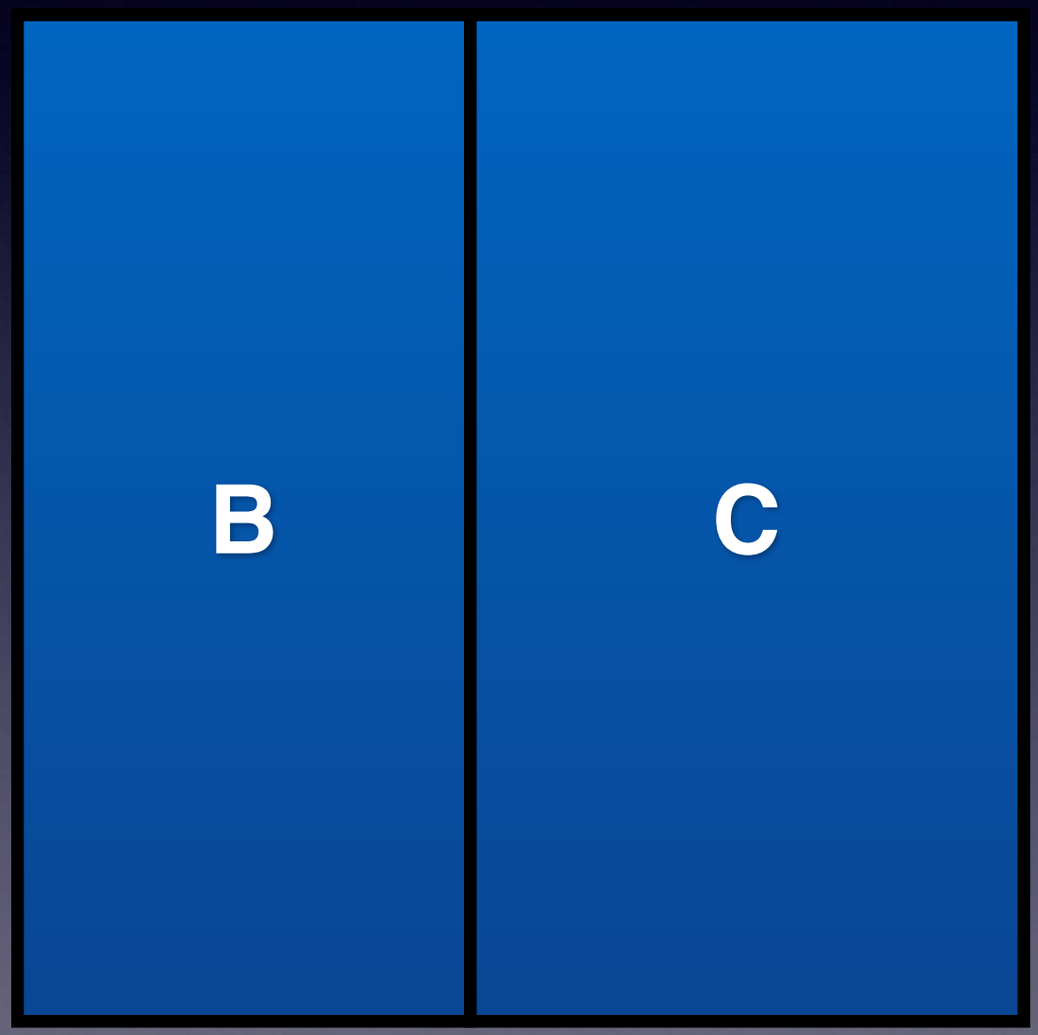
- Space partitioning
- Cellular automata
- Predefined tiles
- Voronoi diagrams
- Delauney triangulation
- Perlin noise
- ...

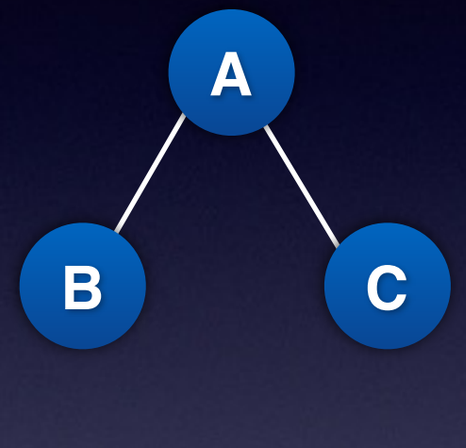
Binary Space Partitioning



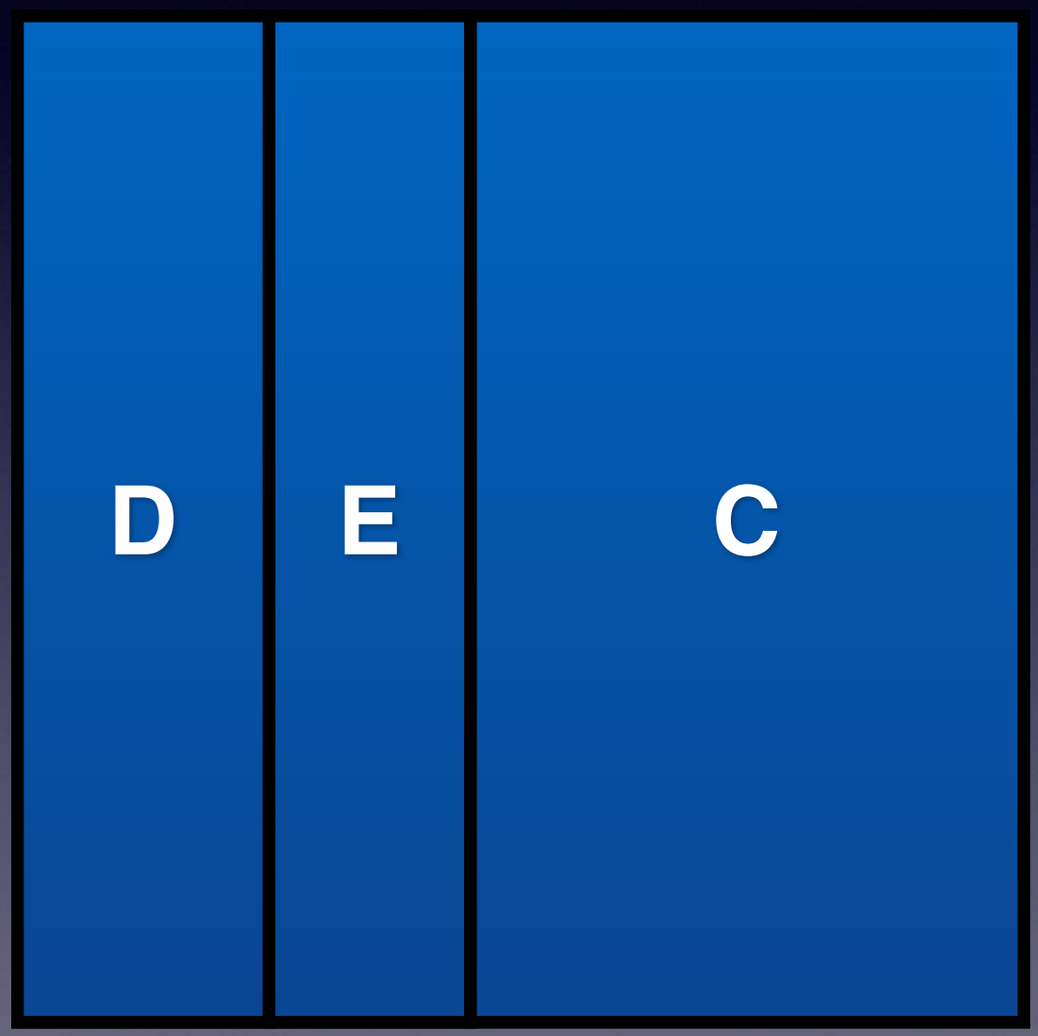
A

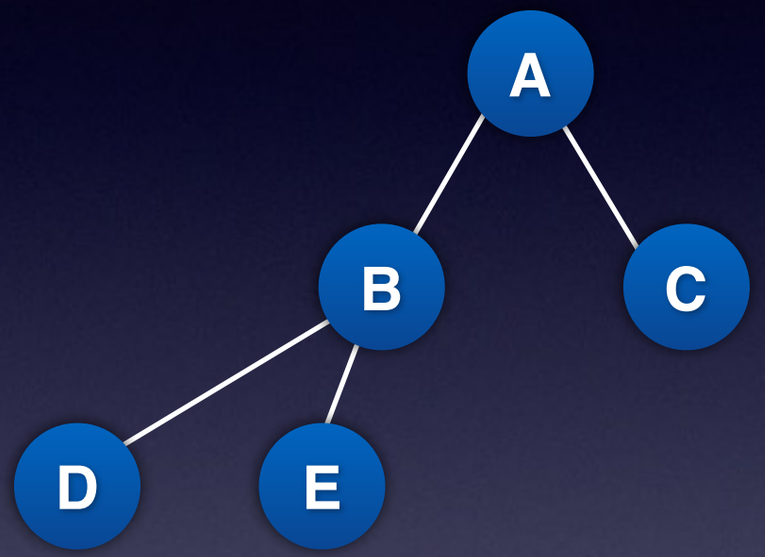
Binary Space Partitioning



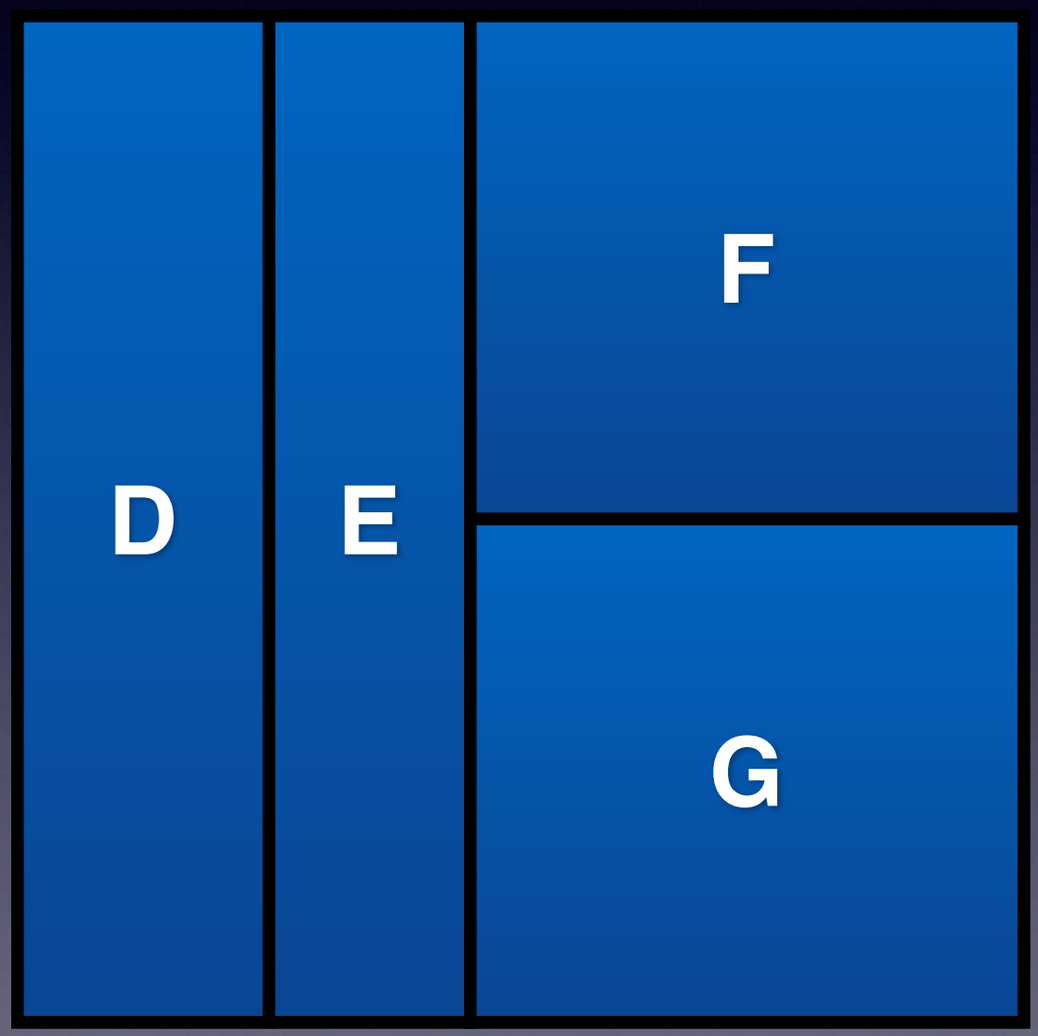


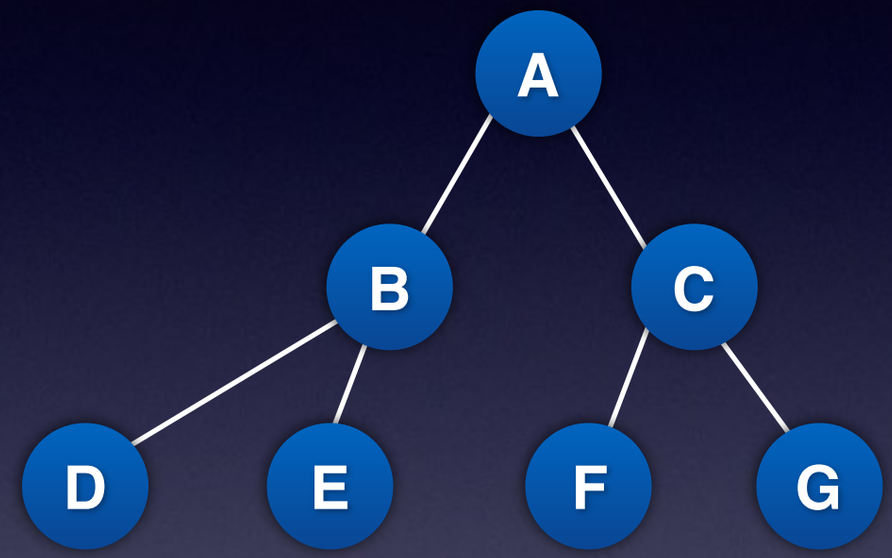
Binary Space Partitioning



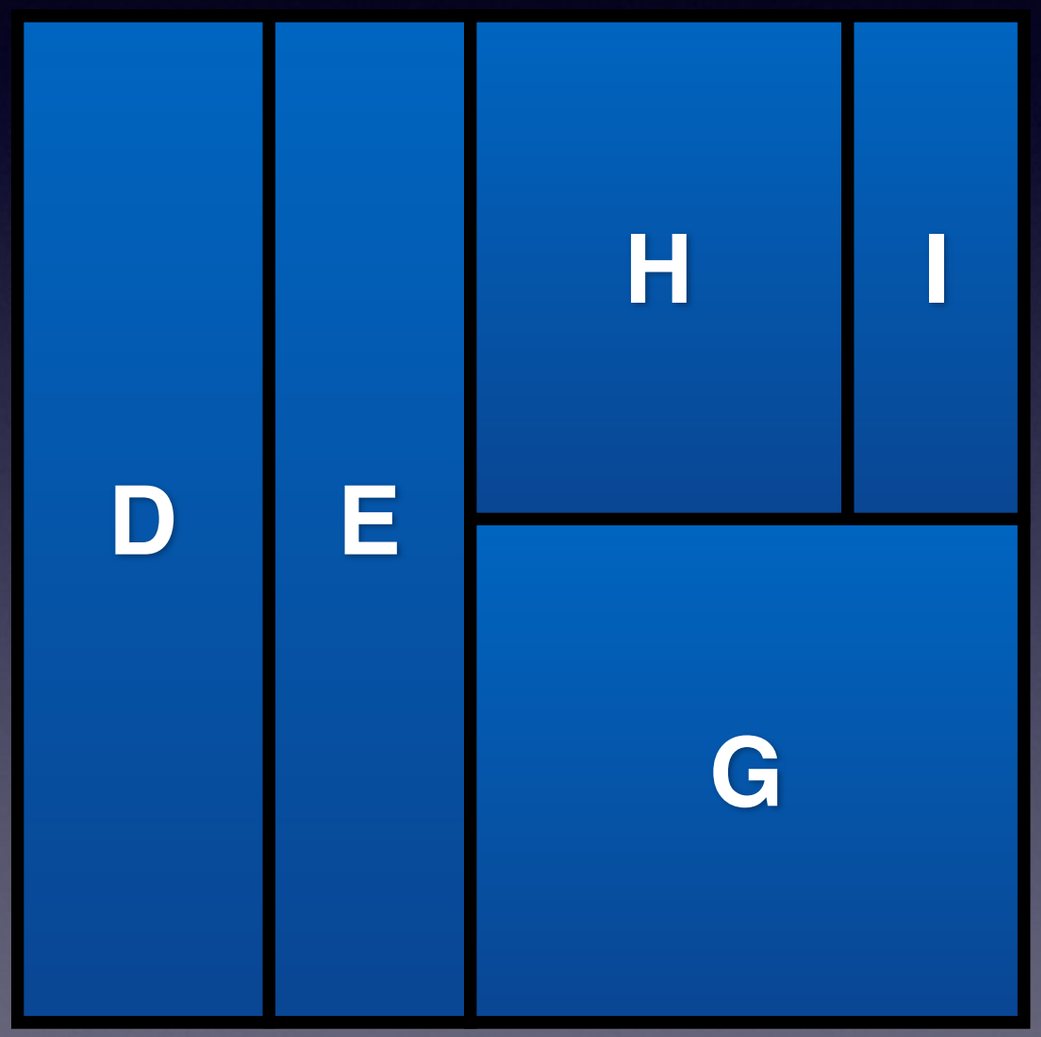


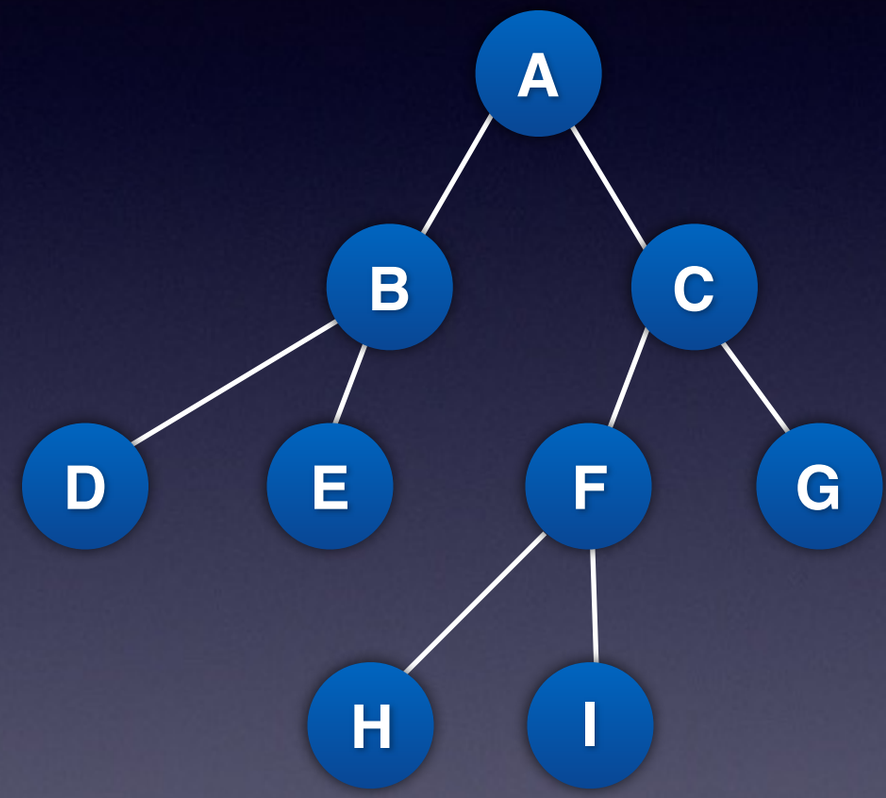
Binary Space Partitioning



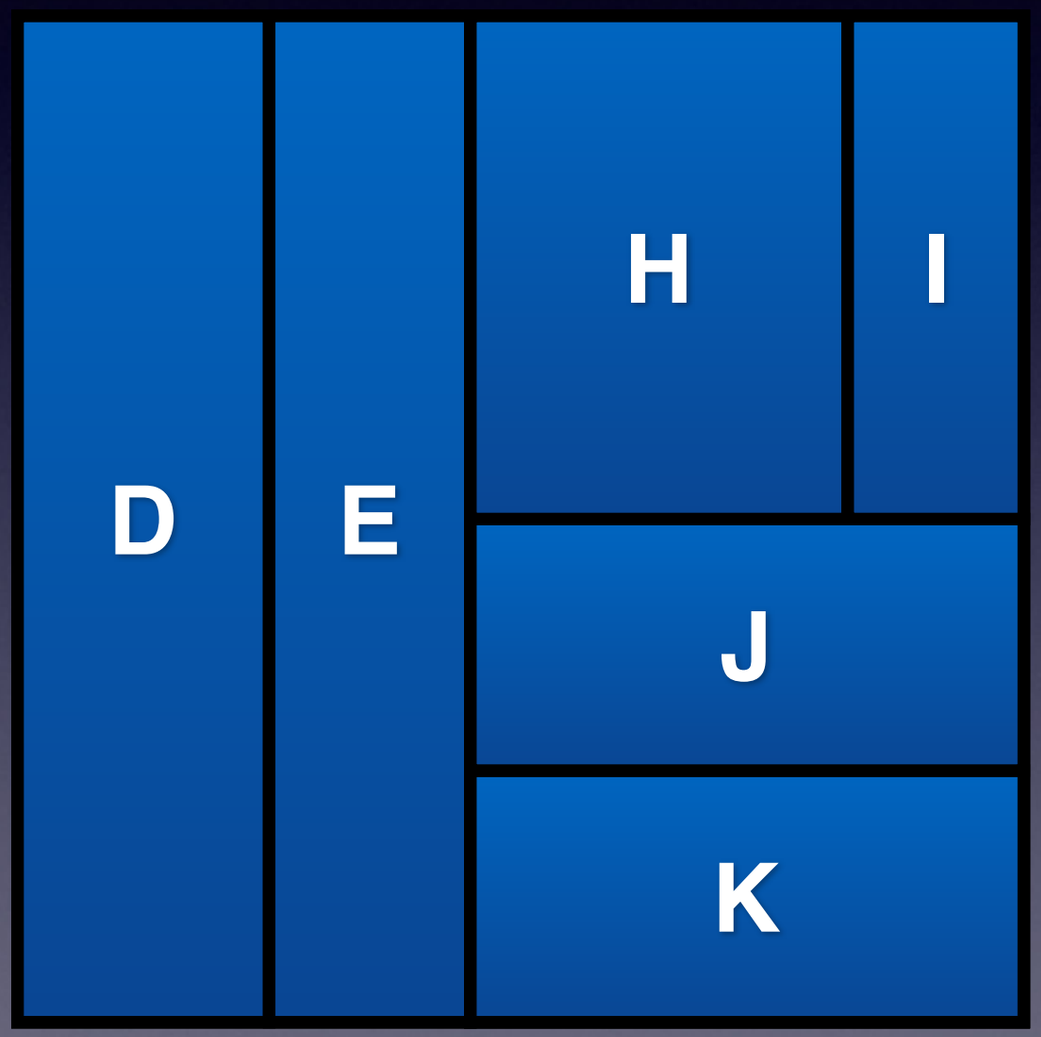


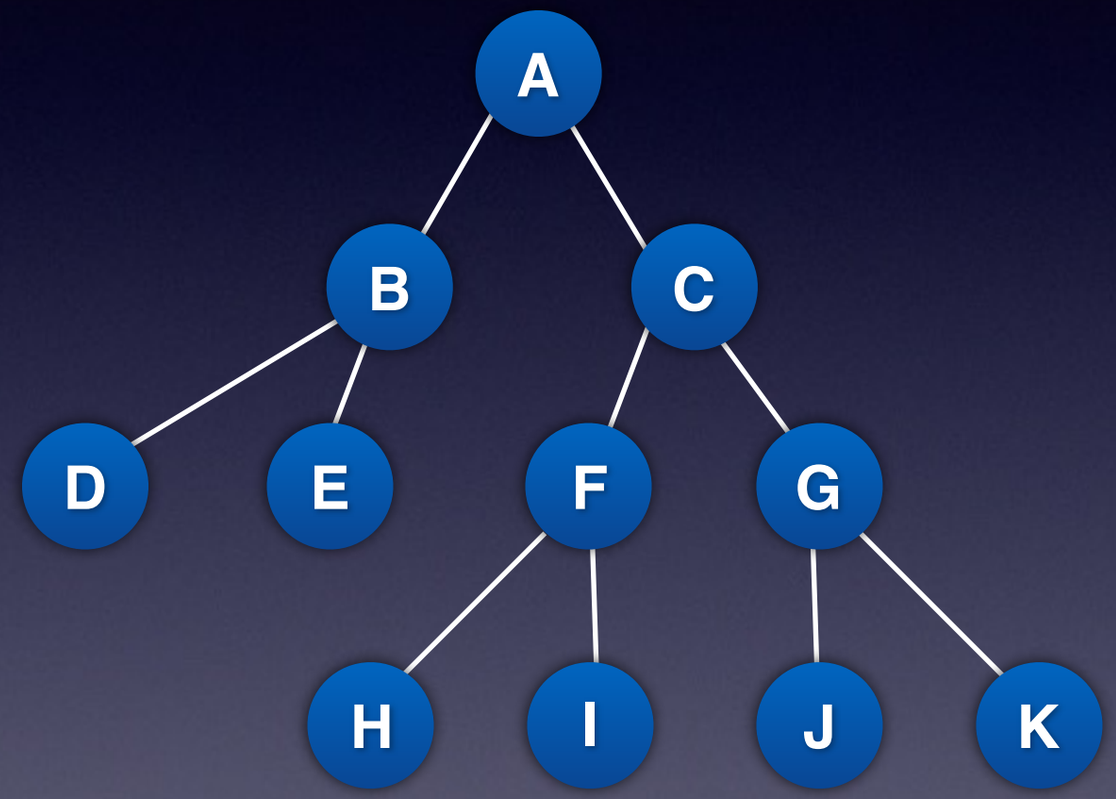
Binary Space Partitioning



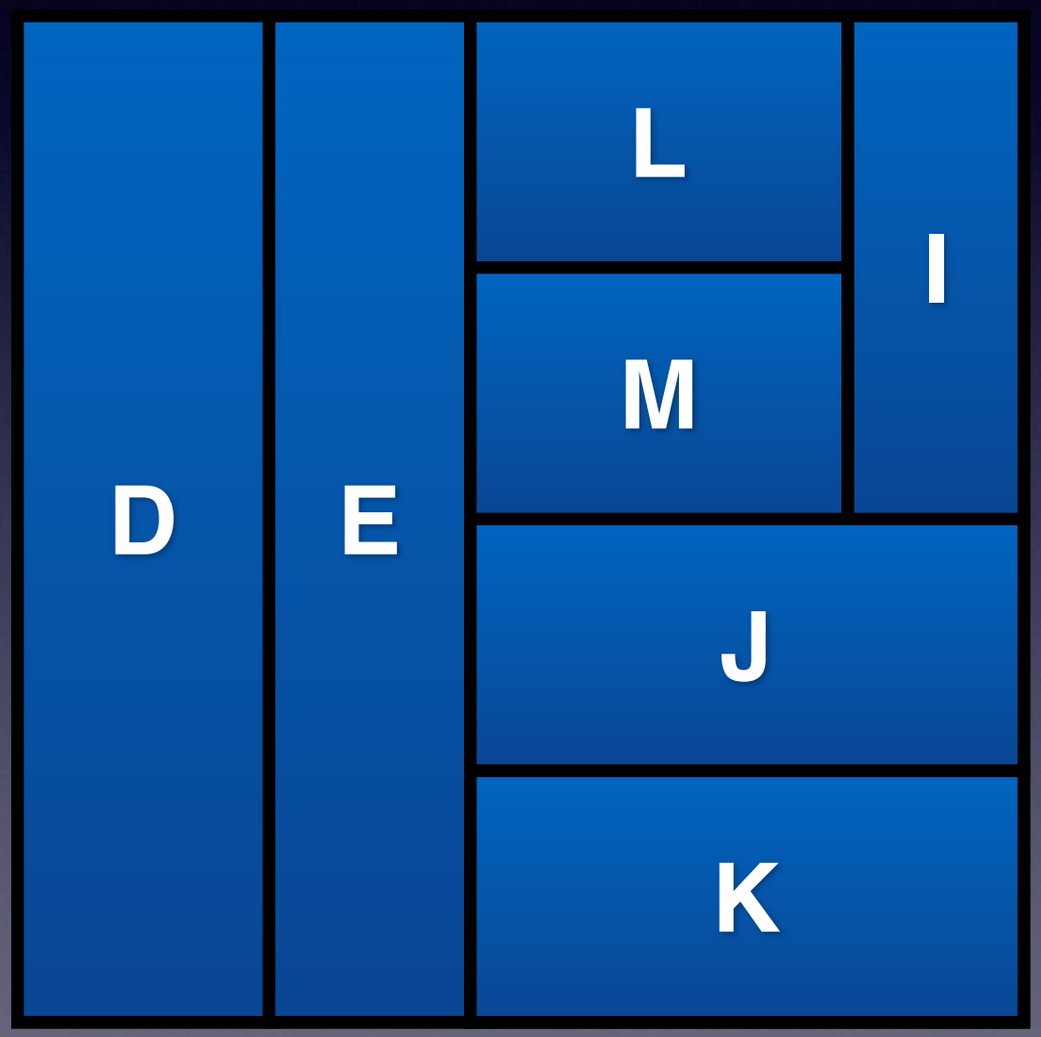


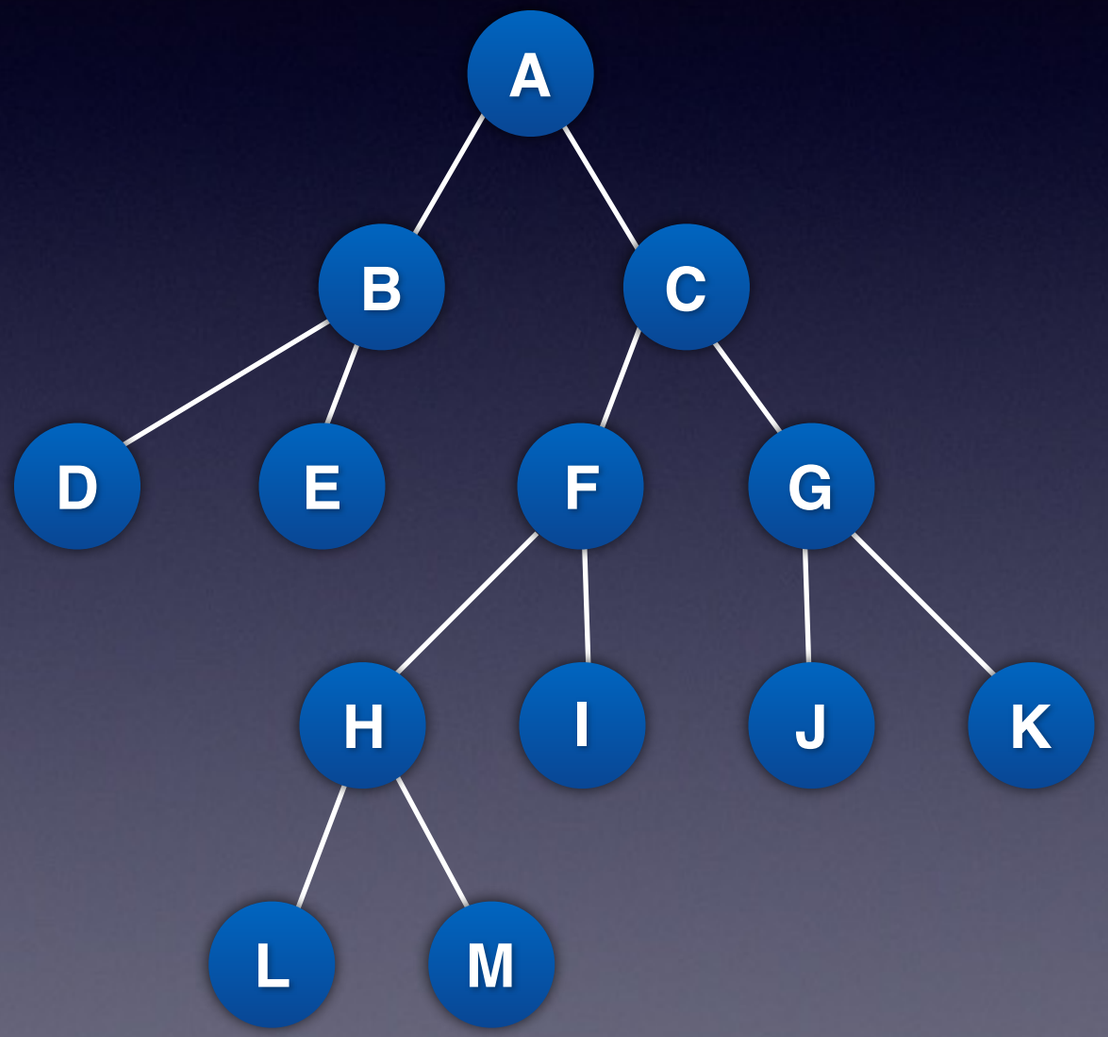
Binary Space Partitioning



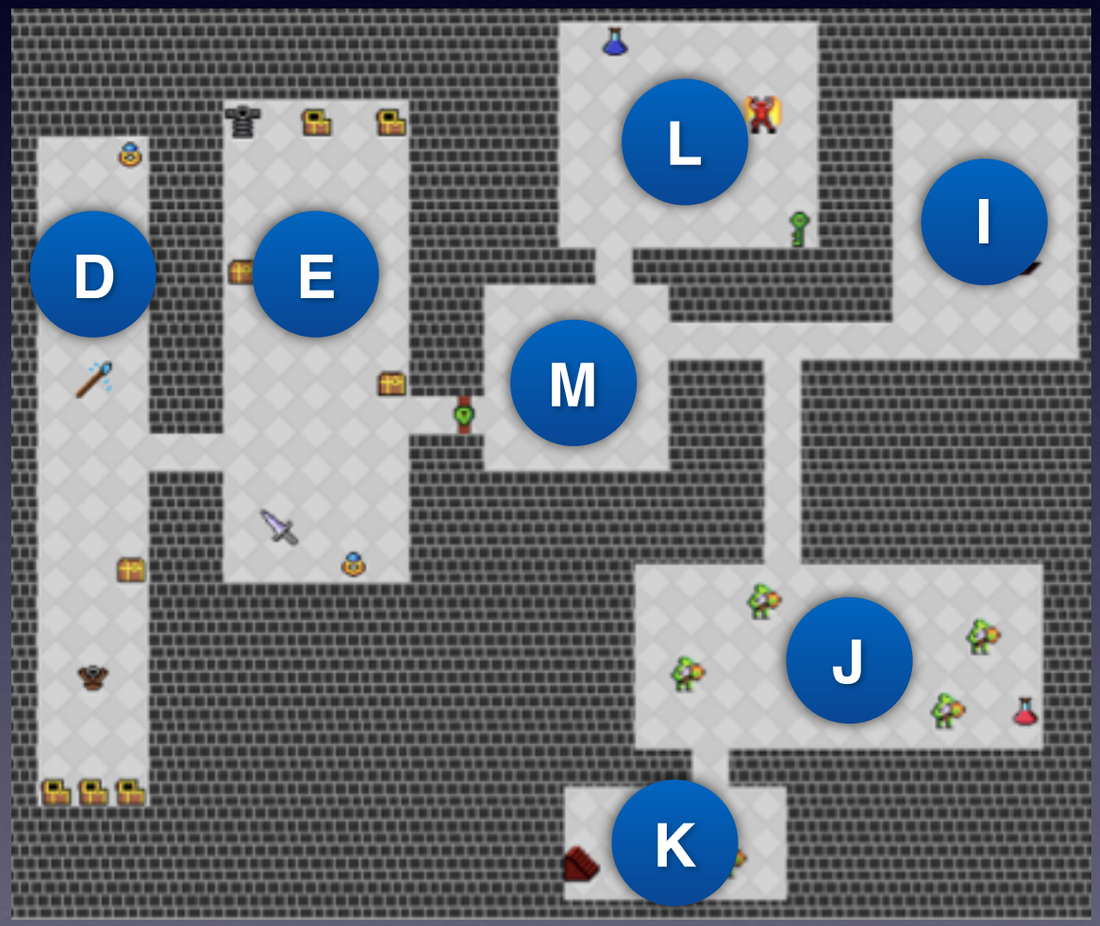


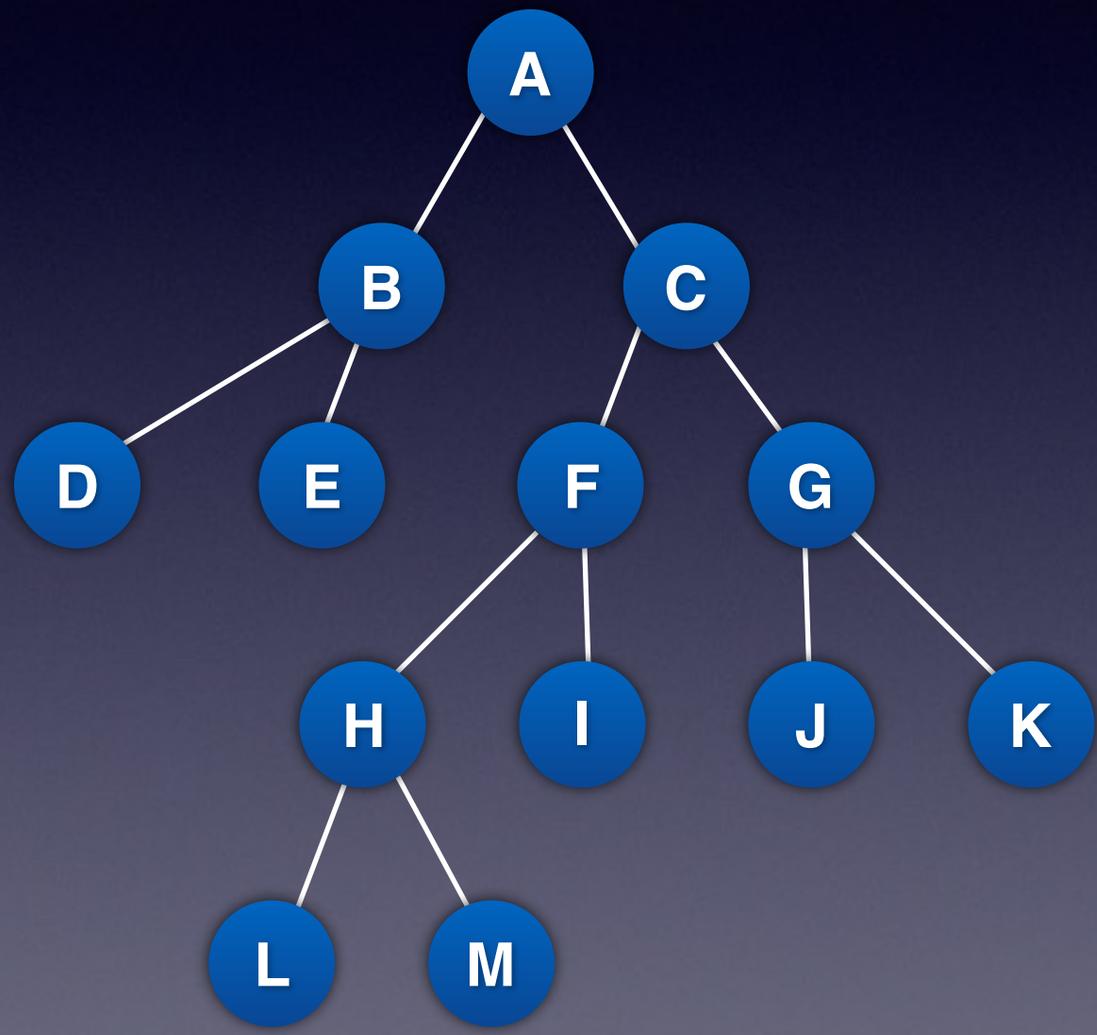
Binary Space Partitioning



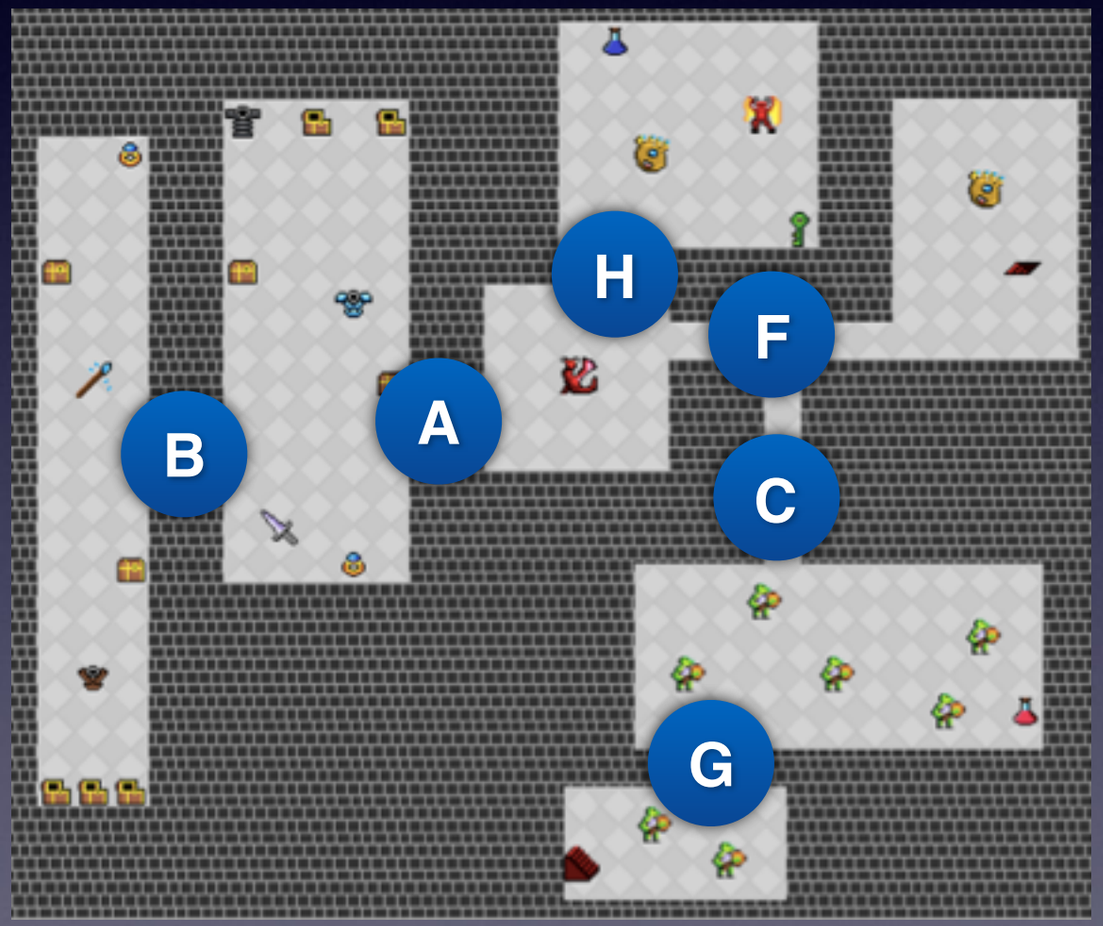


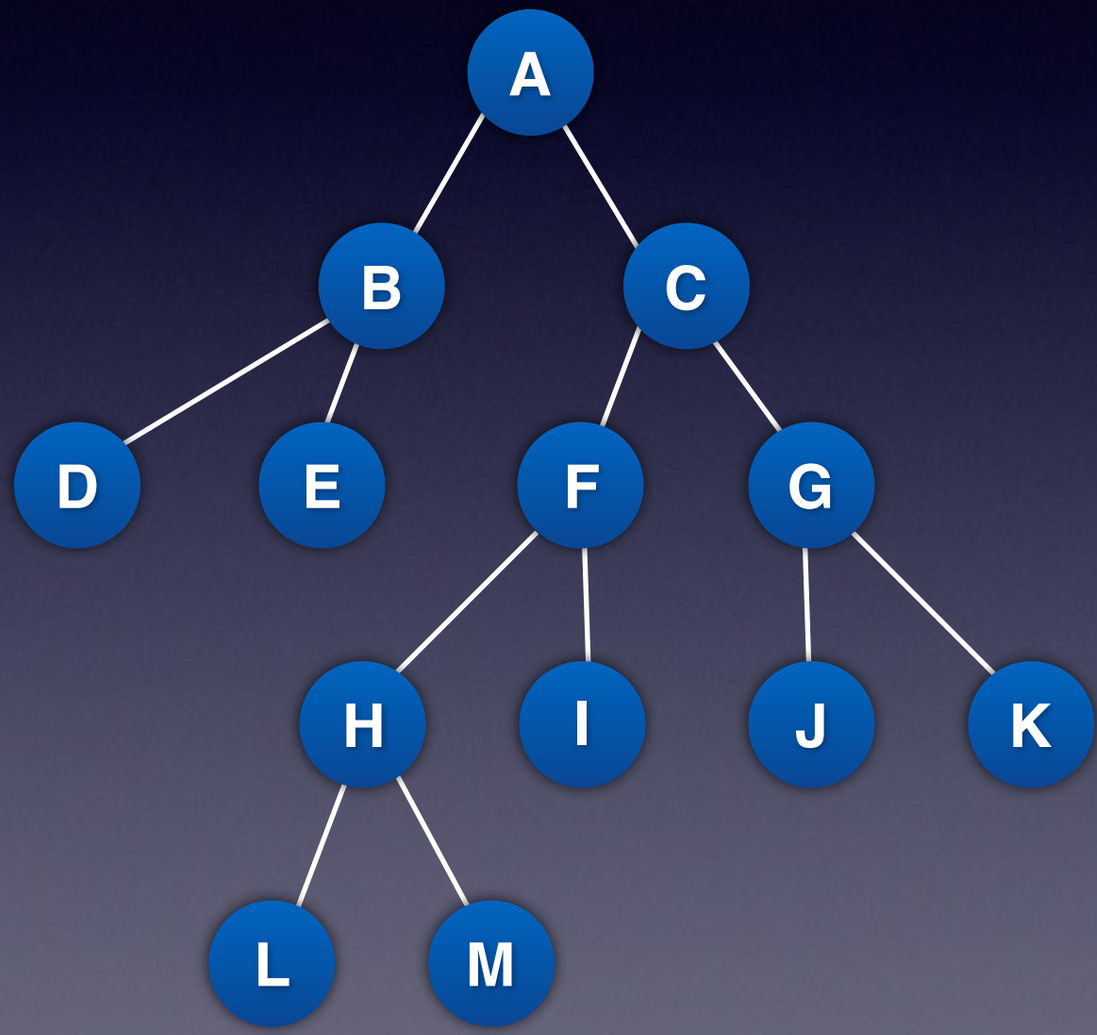
Binary Space Partitioning





Binary Space Partitioning

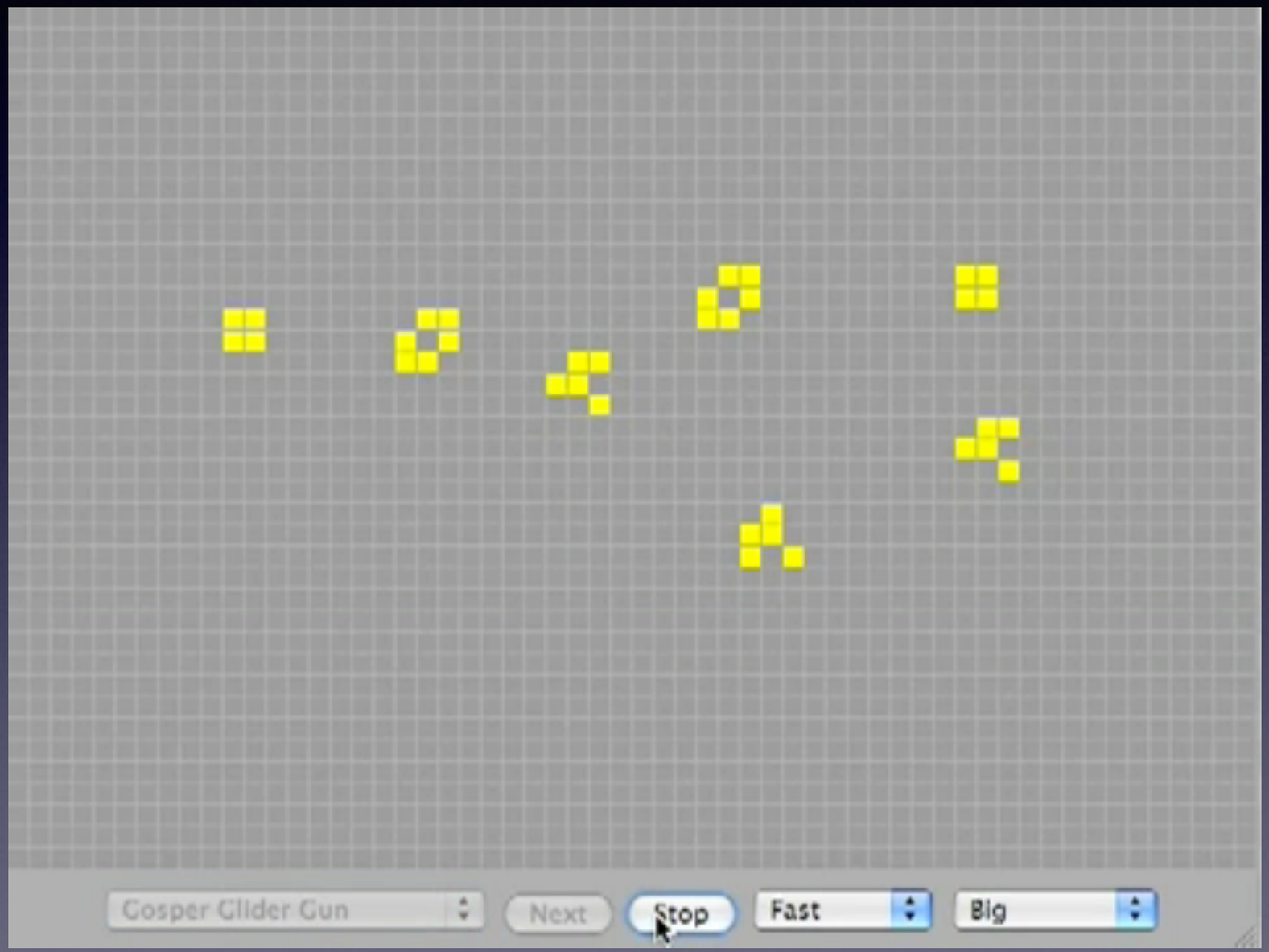




Binary Space Partitioning

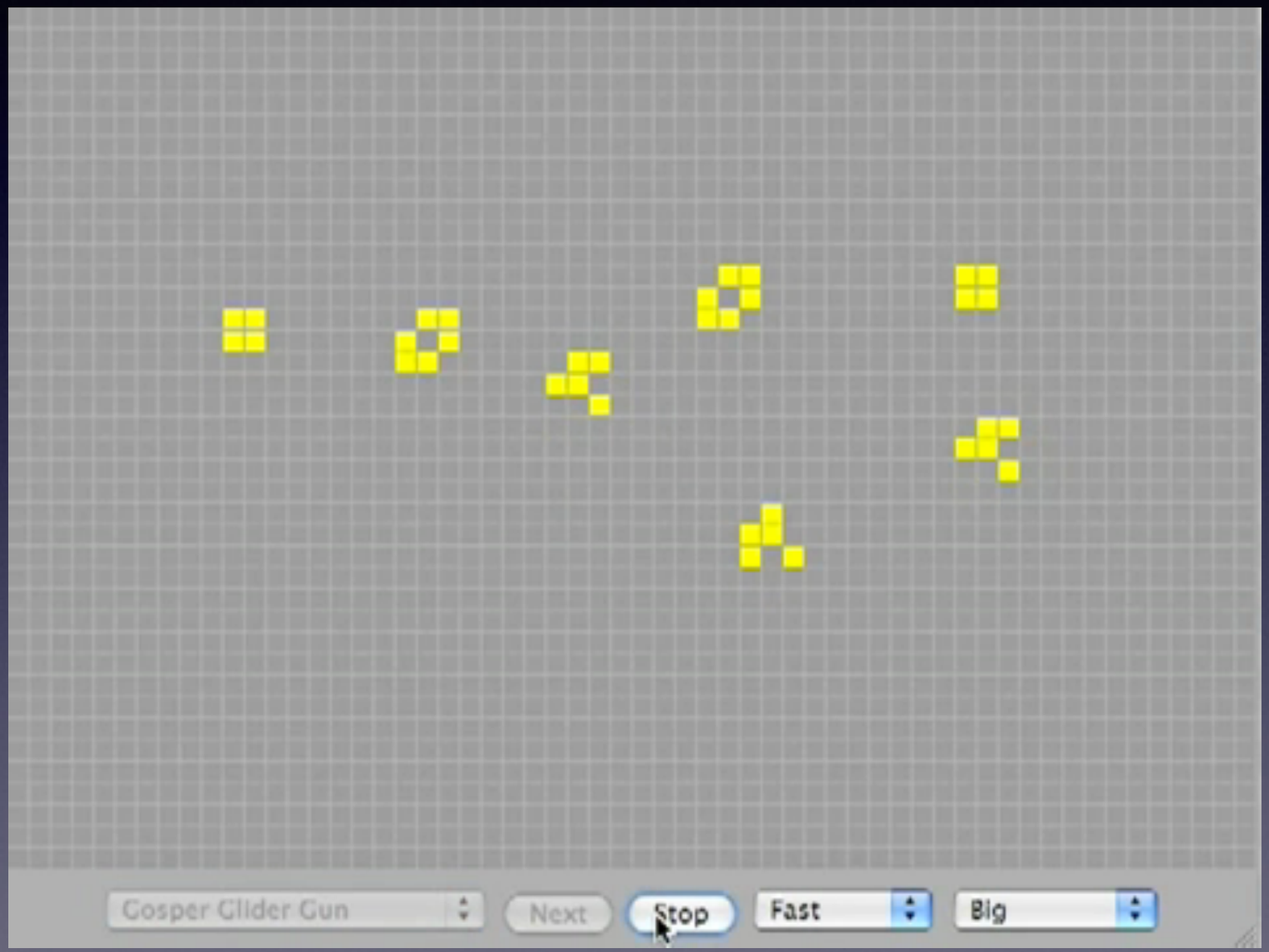
1. Construct a binary partition tree
    - Recursively partition with random lines
    - Stop at minimal acceptable area/dimension
2. Add rooms in leaf areas
    - Draw rooms within leaf areas
    - Connect sections if they share a parent

Cellular Automata



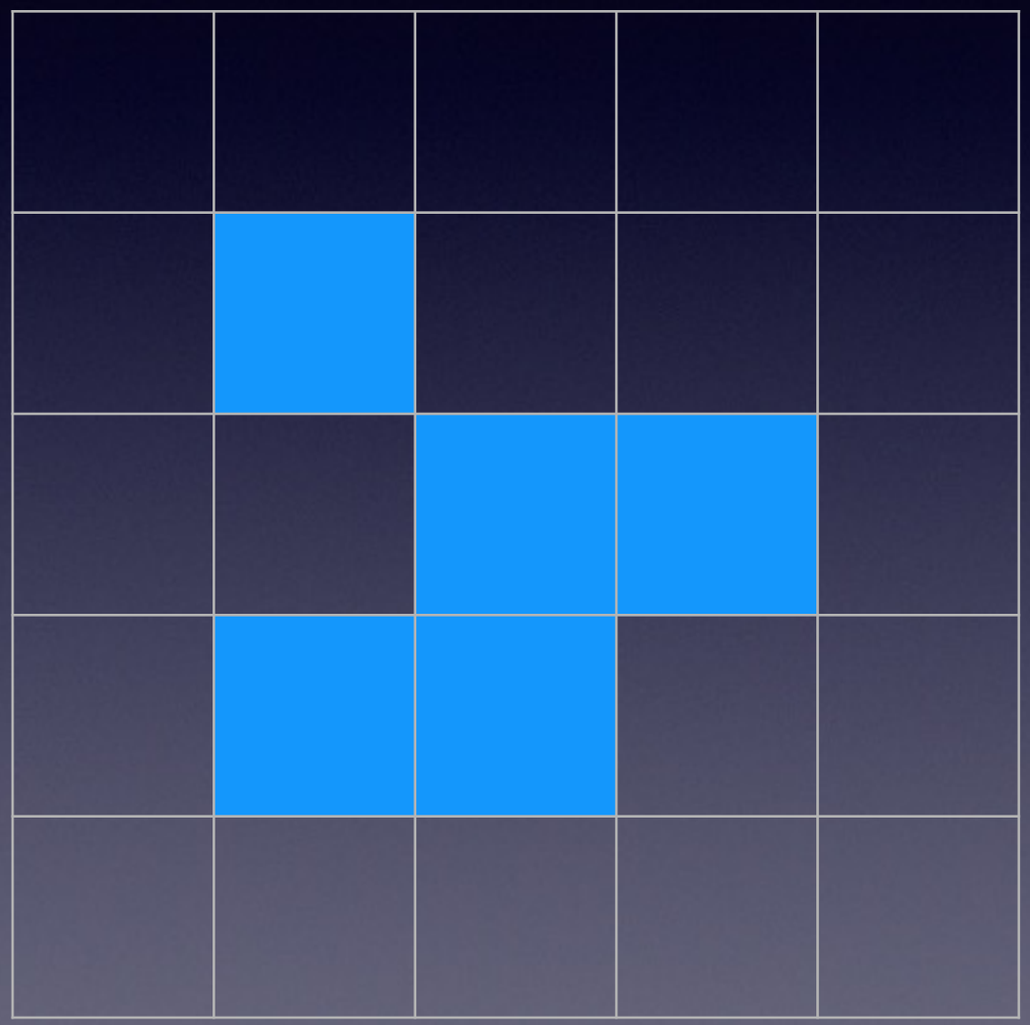
Conway’s Game of Life

Cellular Automata



Conway’s Game of Life

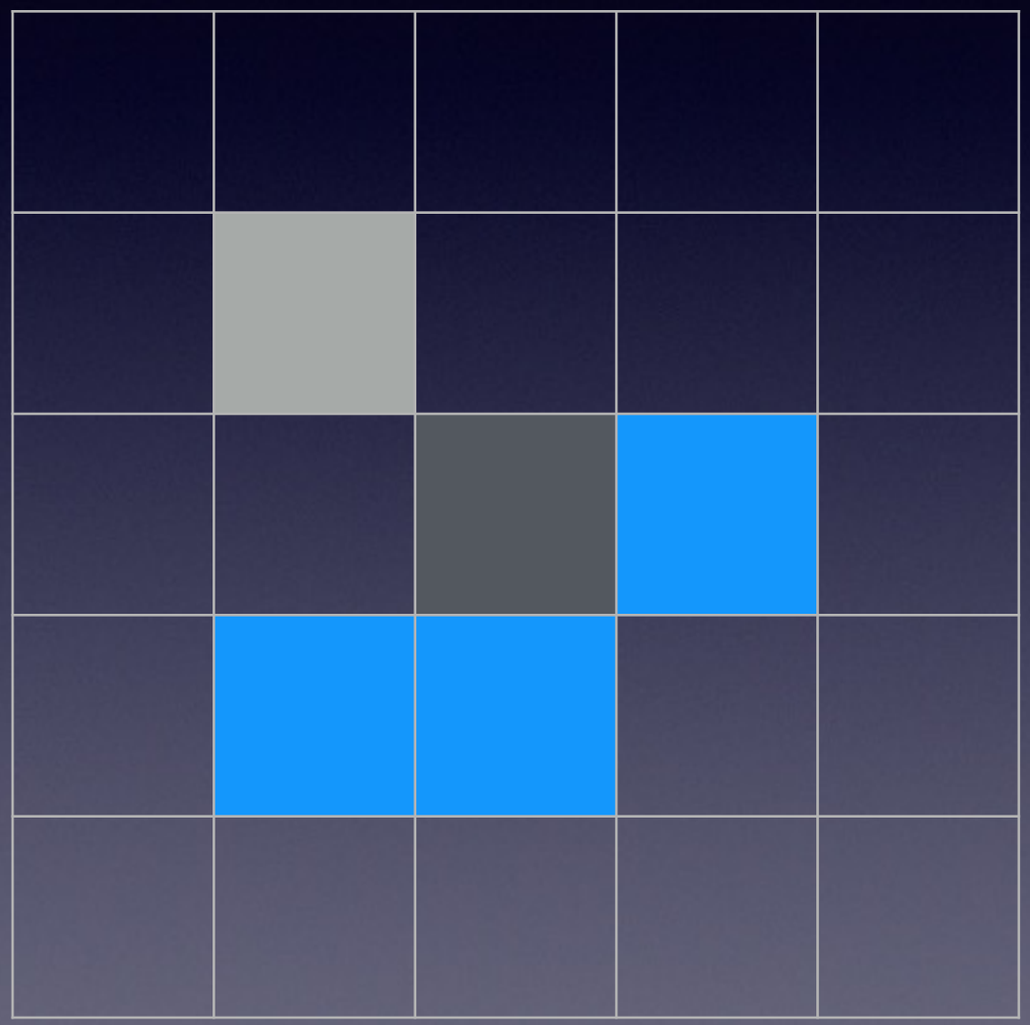
Cellular Automata



Time t

- Stay alive if you have 2-3 neighbours
- Come alive if you have 3 neighbours

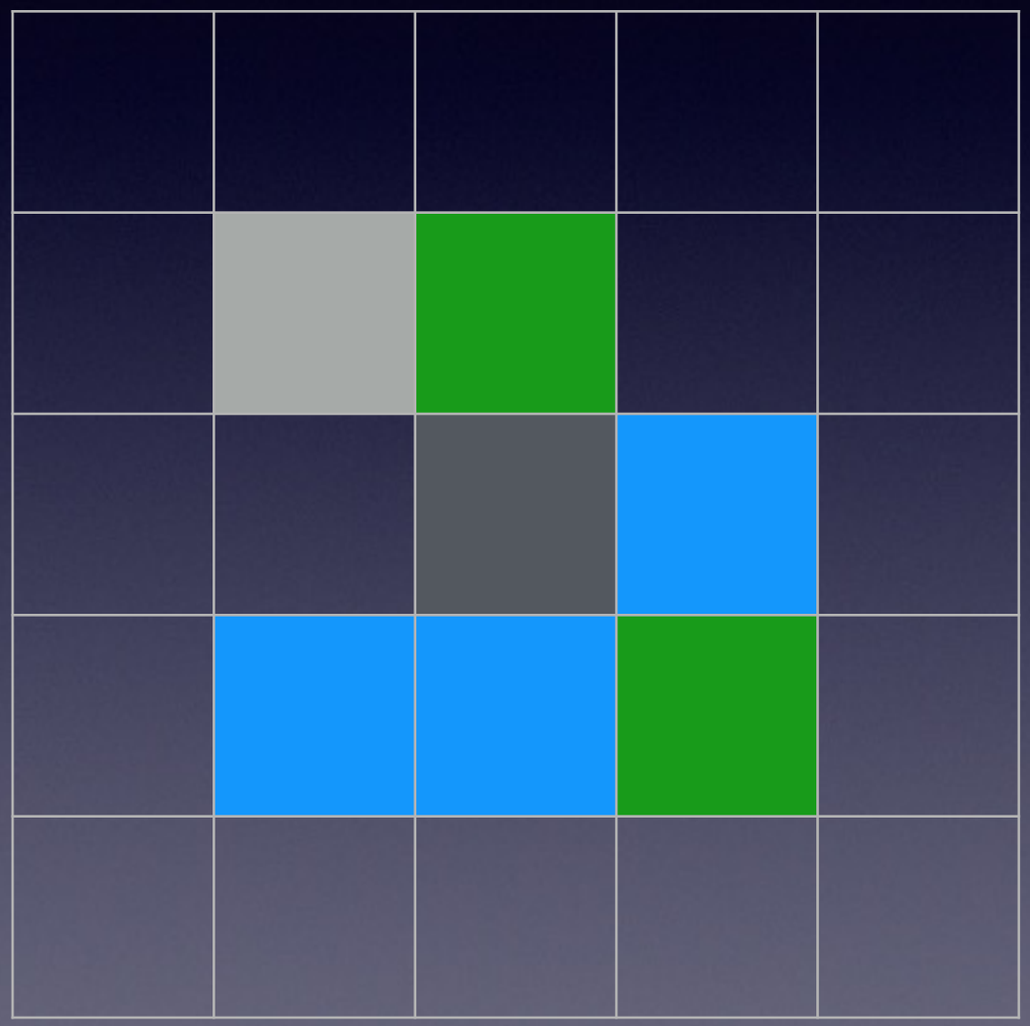
Cellular Automata



Time t

- Stay alive if you have 2-3 neighbours
- Come alive if you have 3 neighbours

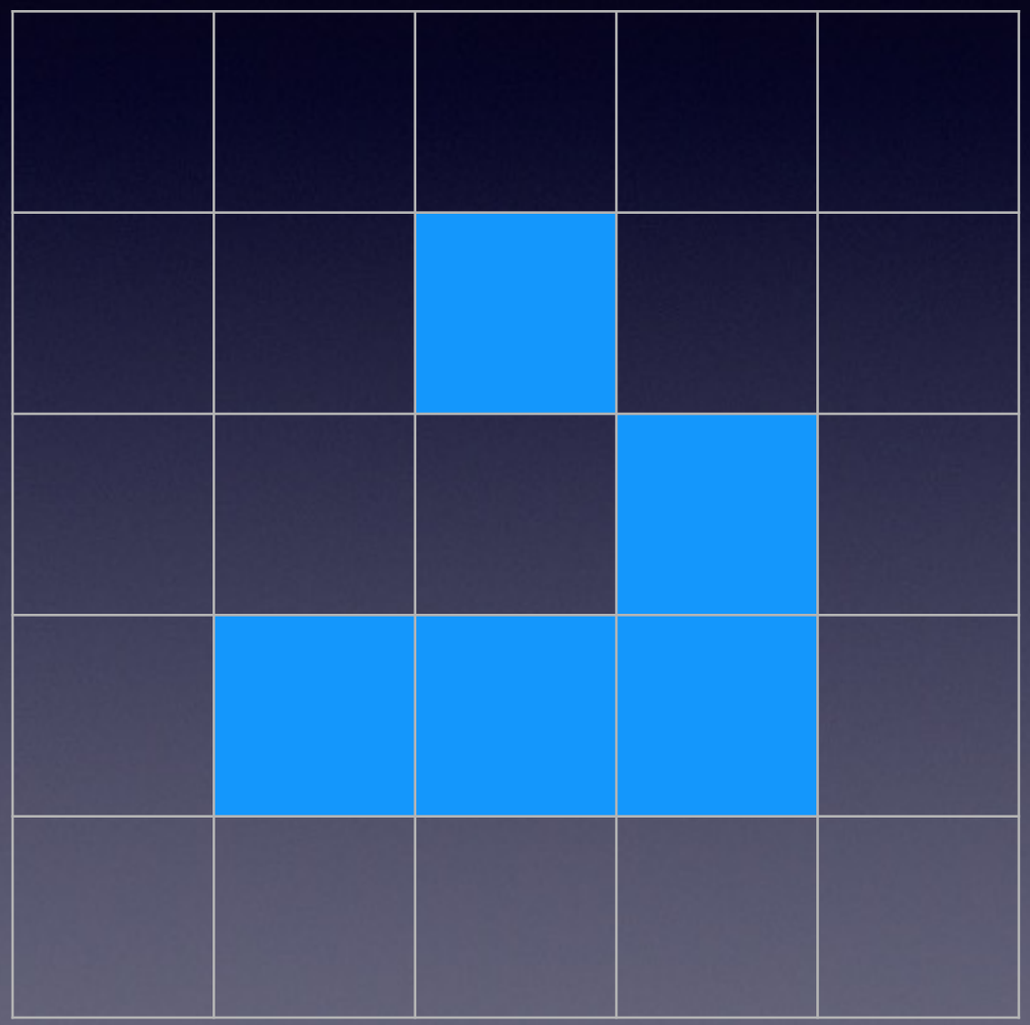
Cellular Automata



Time t

- Stay alive if you have 2-3 neighbours
- Come alive if you have 3 neighbours

Cellular Automata



Time t+1

- Stay alive if you have 2-3 neighbours
- Come alive if you have 3 neighbours

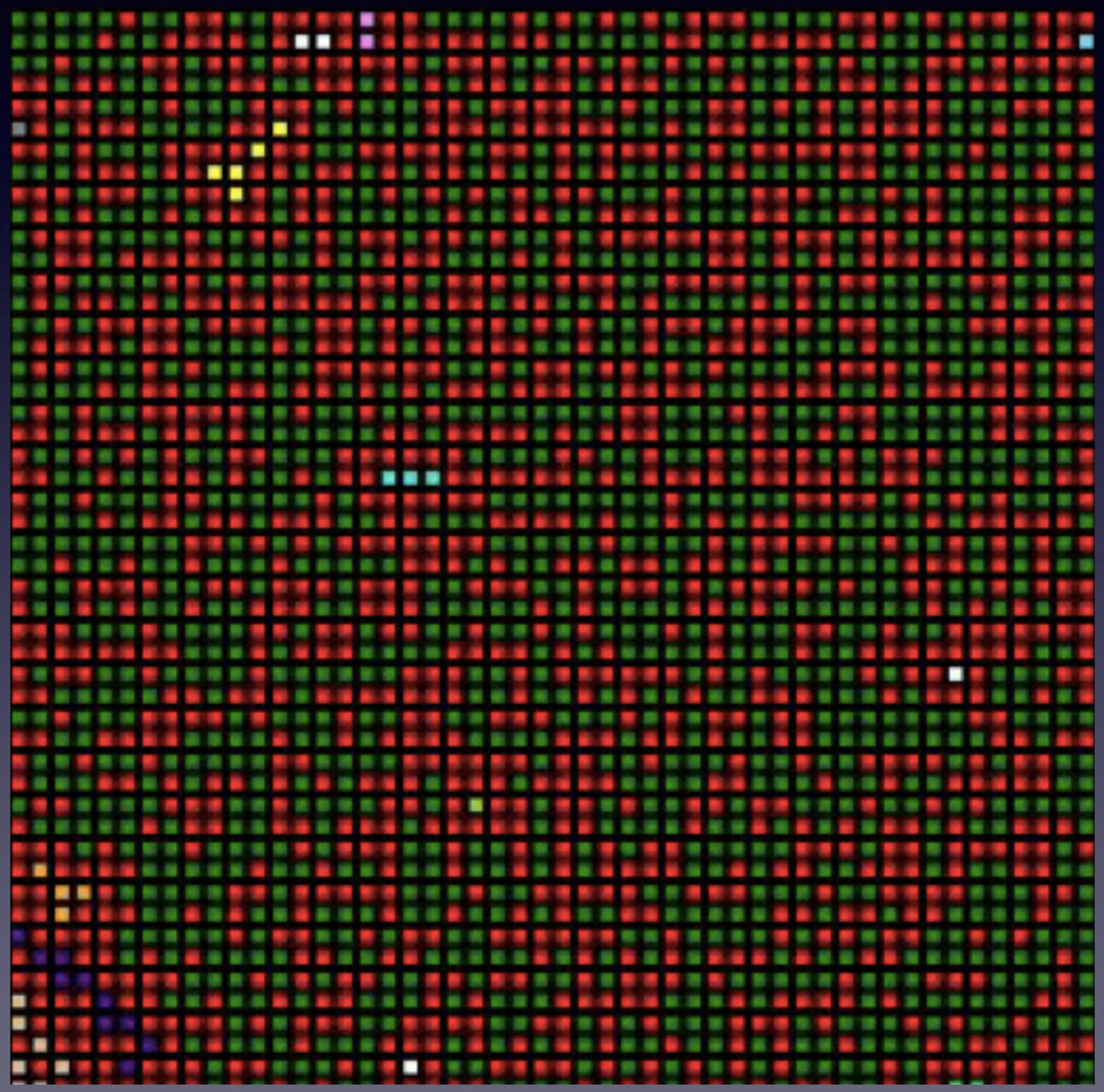
Cellular Automata

Cave Generation

- Johnson et al. (2010) use a 50x50 grid with 2 states: **empty** and **rock**
- Initial uniform distribution of rock (p=0.5)
- Automata runs for k steps
- Rock if at least T rock neighbours, else empty

Cellular Automata

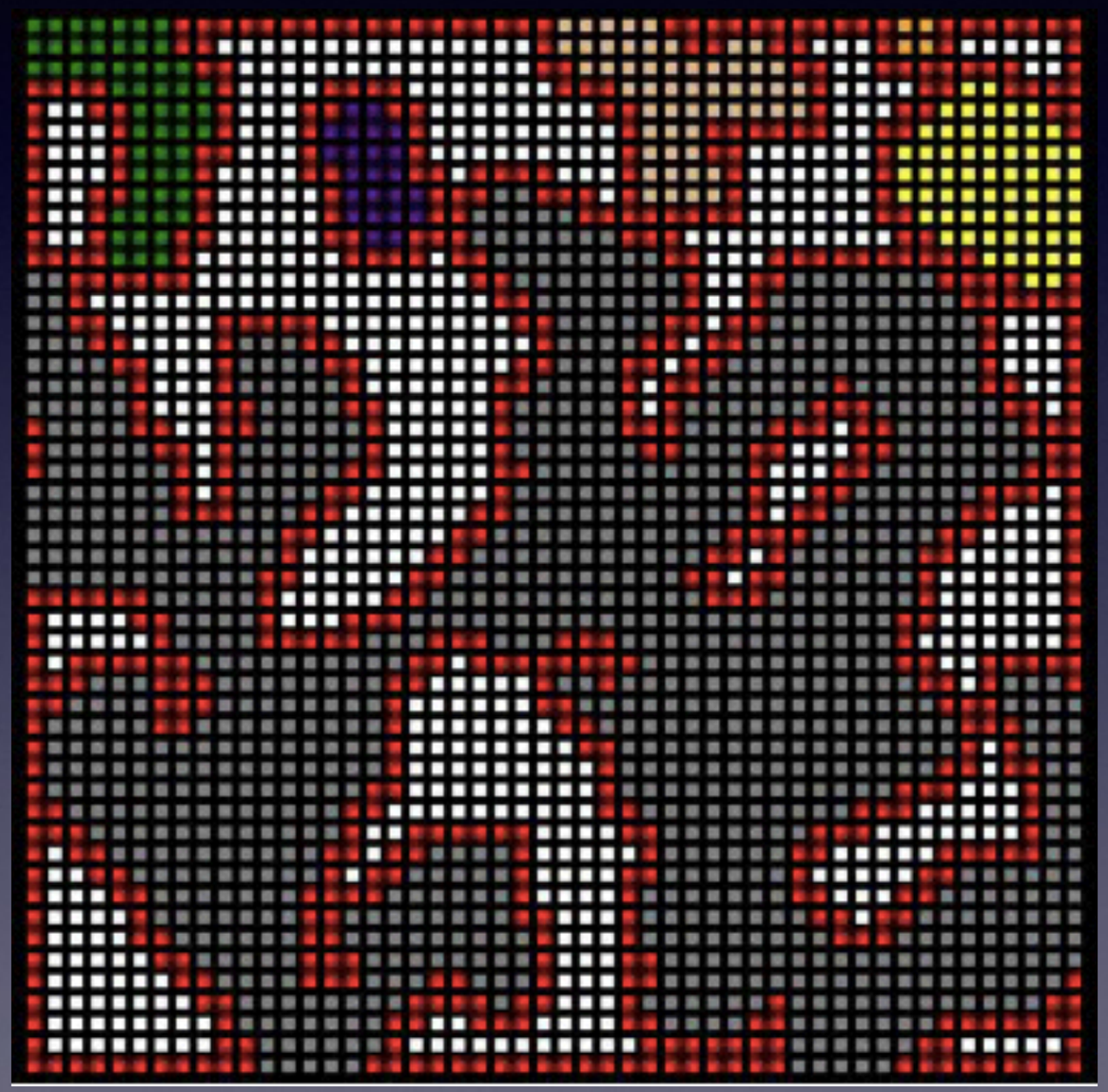
Cave Generation



Initial grid (red/white=rock, other colours=caves)

Cellular Automata

Cave Generation



4 steps, T=5 (red/white=rock, other colours=caves)

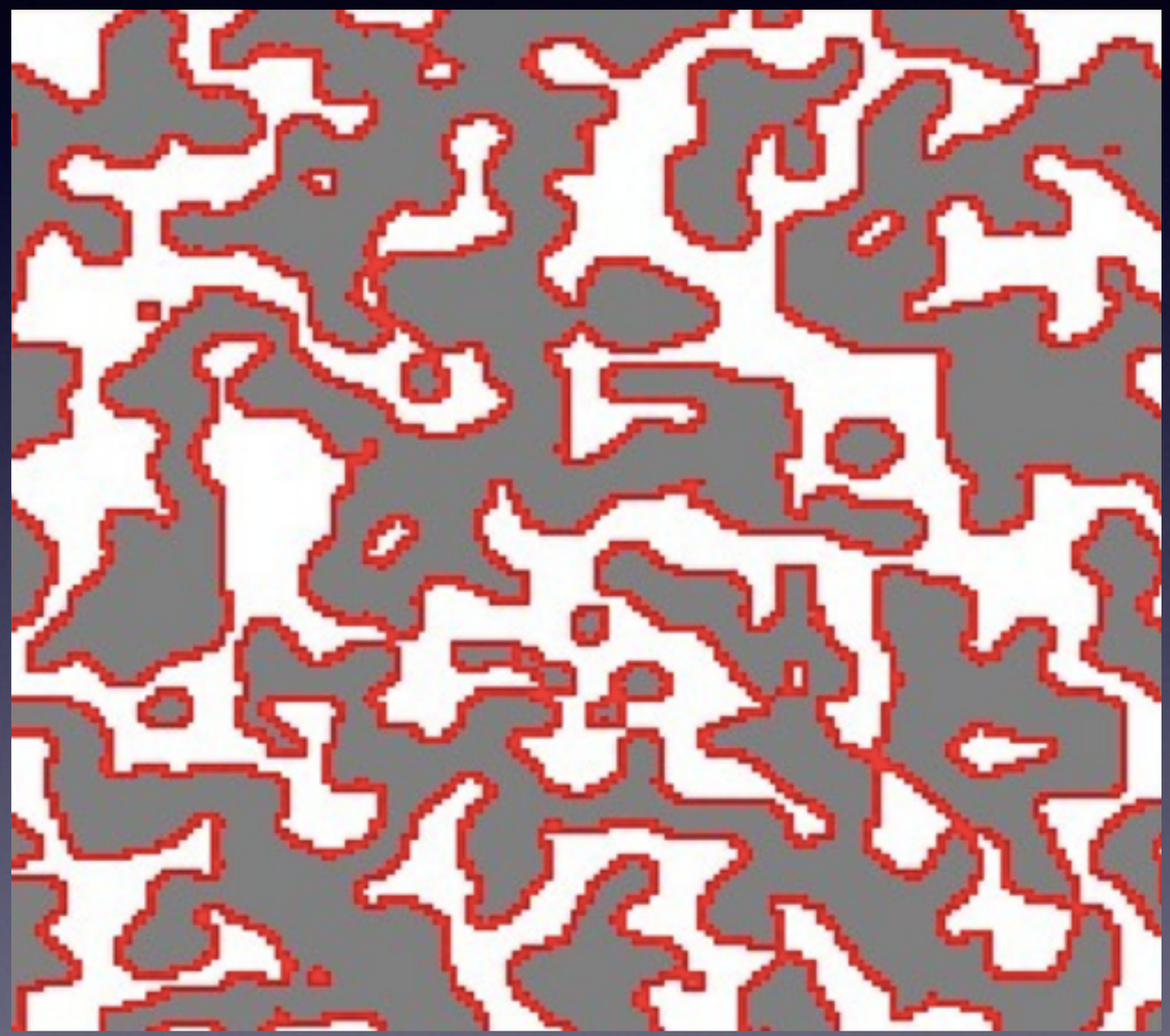
Cellular Automata

Cave Networks

- CA creates one or more caves
- Johnson et al. (2010) generate a set of CA tiles:
    - Check pairs of bordering tiles: are the largest caves connected?
    - If not, **drill** at point of least separation.
    - Run CA for 2 steps to smooth edges.
- Tiles generated on-the-fly to give infinite caves.

Cellular Automata

Cave Networks



A 3x3 grid of CA tiles, with T=13

Spleunky (2008)

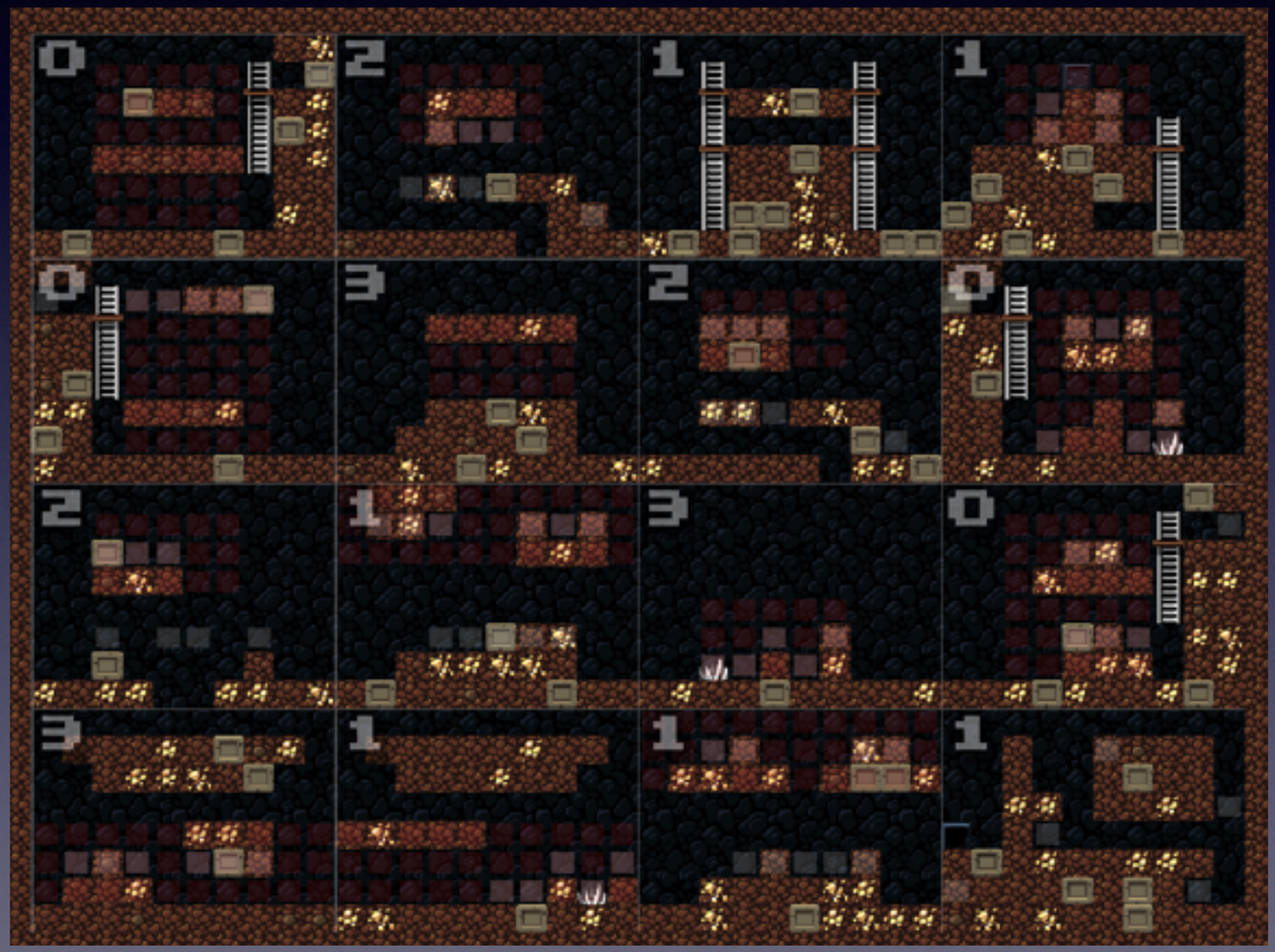
Tile-based Generation

Spleunky (2008)

Tile-based Generation

Spleunky (2008)

Tile-based Generation



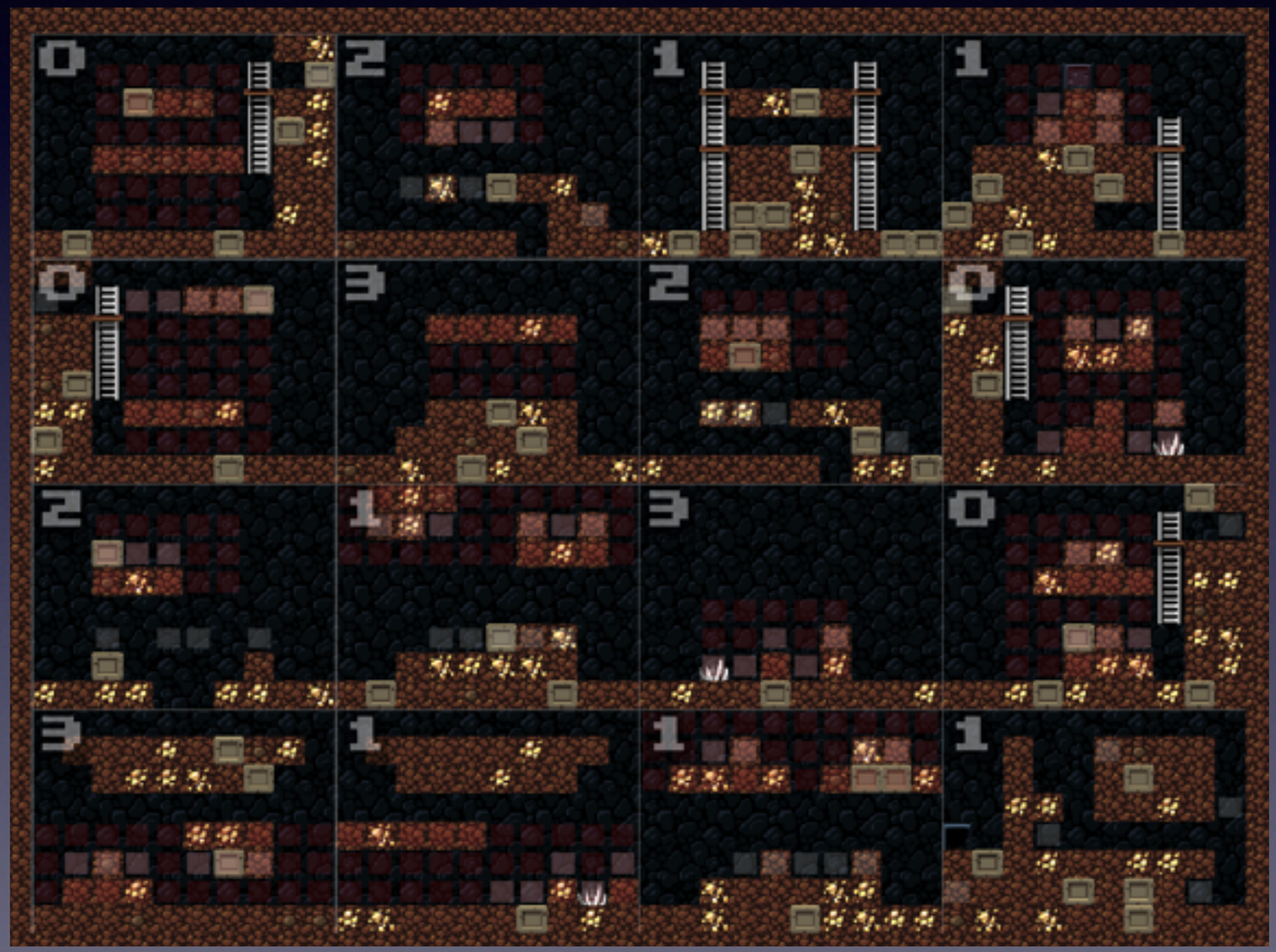
Spleunky (2008)

Tile-based Generation

Construct 4x4 grid of room types, with path from start to exit.

Select room templates

Select probabilistic tile types.



Sir, You Are Being Hunted

(Big Robot 2013)

The British Countryside Generator

Sir, You Are Being Hunted

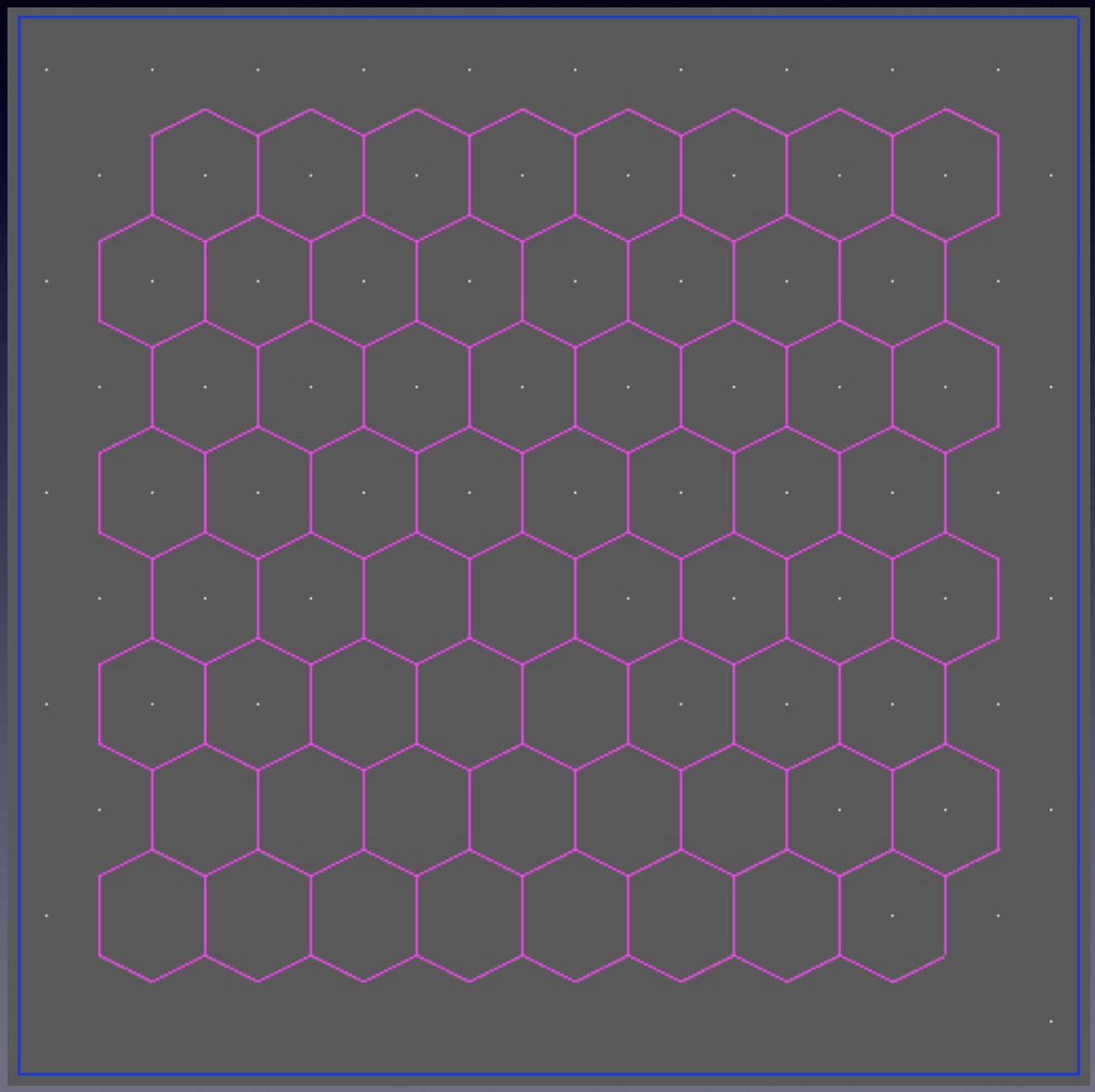
(Big Robot 2013)

The British Countryside Generator

Sir, You Are Being Hunted

Tile-based Generation

Each island starts
as a hexagonal grid



Sir, You Are Being Hunted

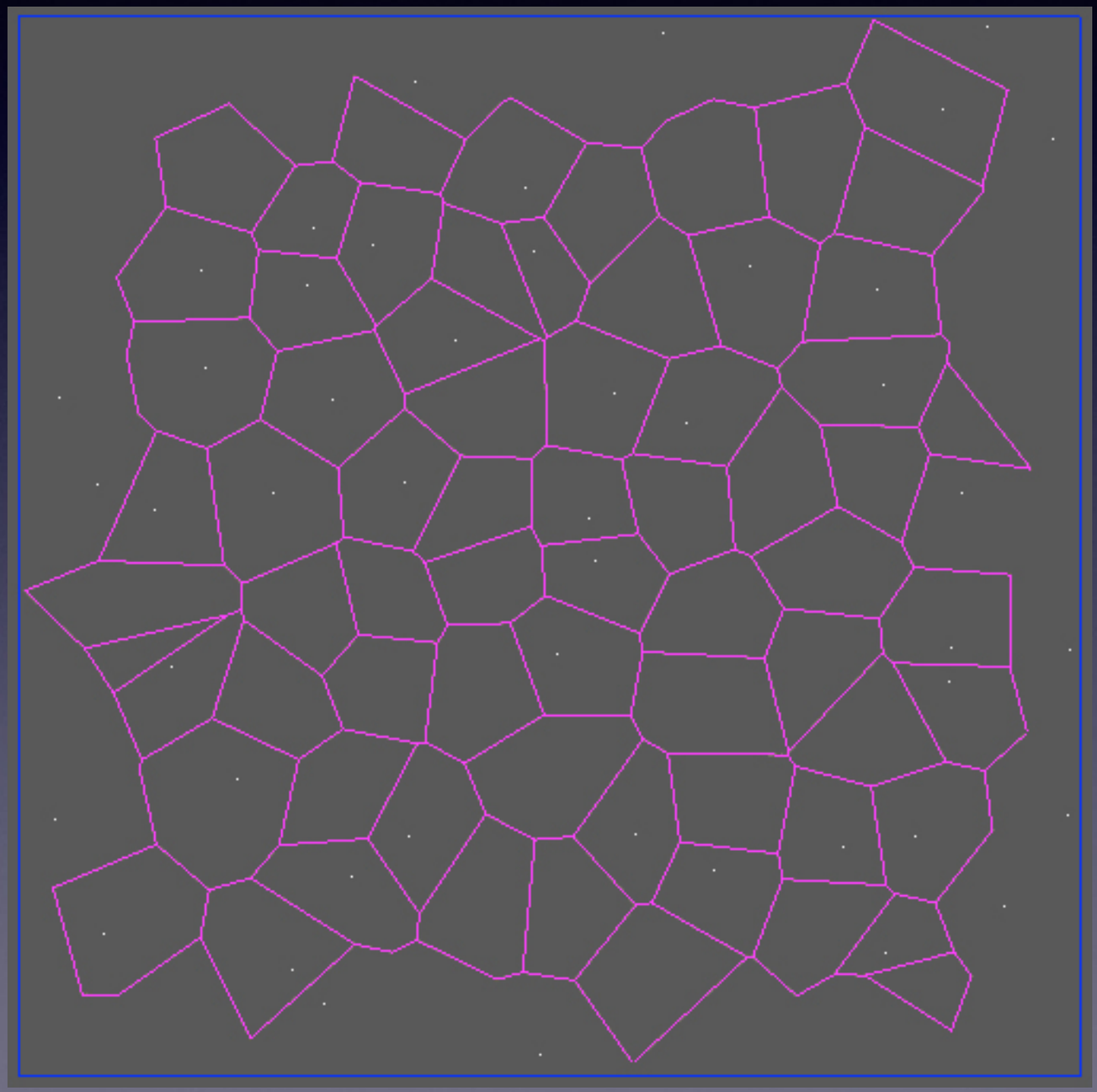
Tile-based Generation

Disturb grid:
randomly displace
centre points

Voronoi diagram
defines tiles

Perlin noise for tile
height.

Predefined tile
themes.



Sir, You Are Being Hunted

Disturbed Grid Tiles

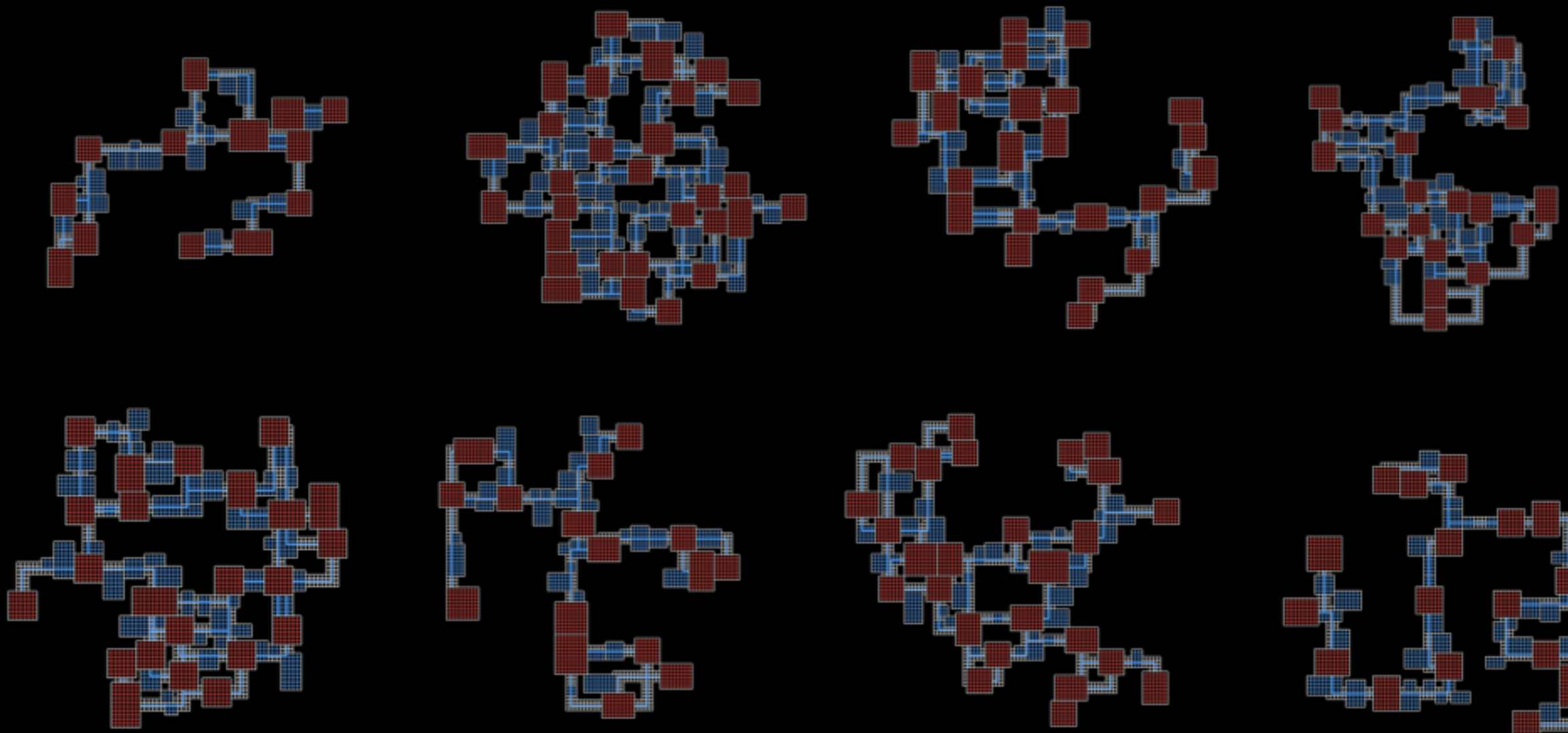
TinyKeep
(PhiGames 2014)

TinyKeep
(PhiGames 2014)

TinyKeep
(PhiGames 2014)

TinyKeep

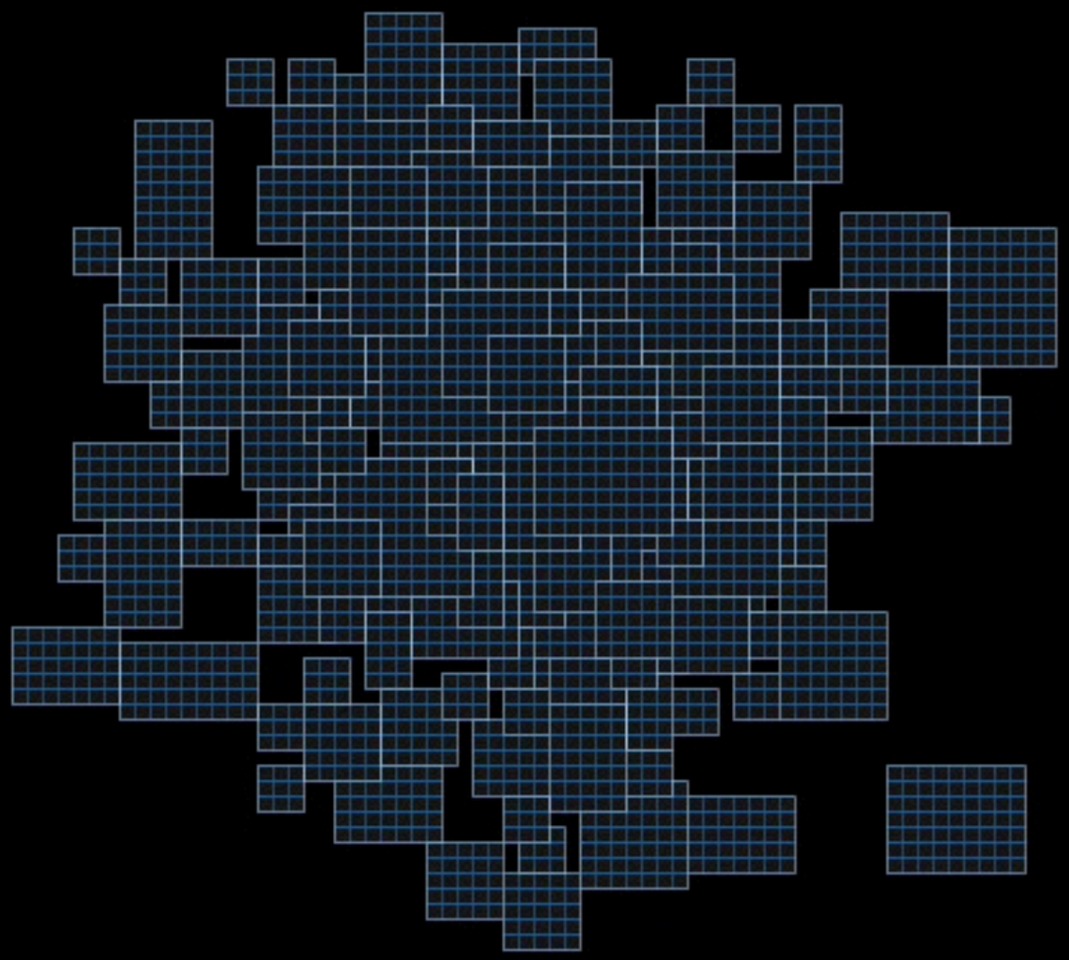
Dungeon Generator



http://tinykeep.com/dungen/

TinyKeep

Dungeon Generator

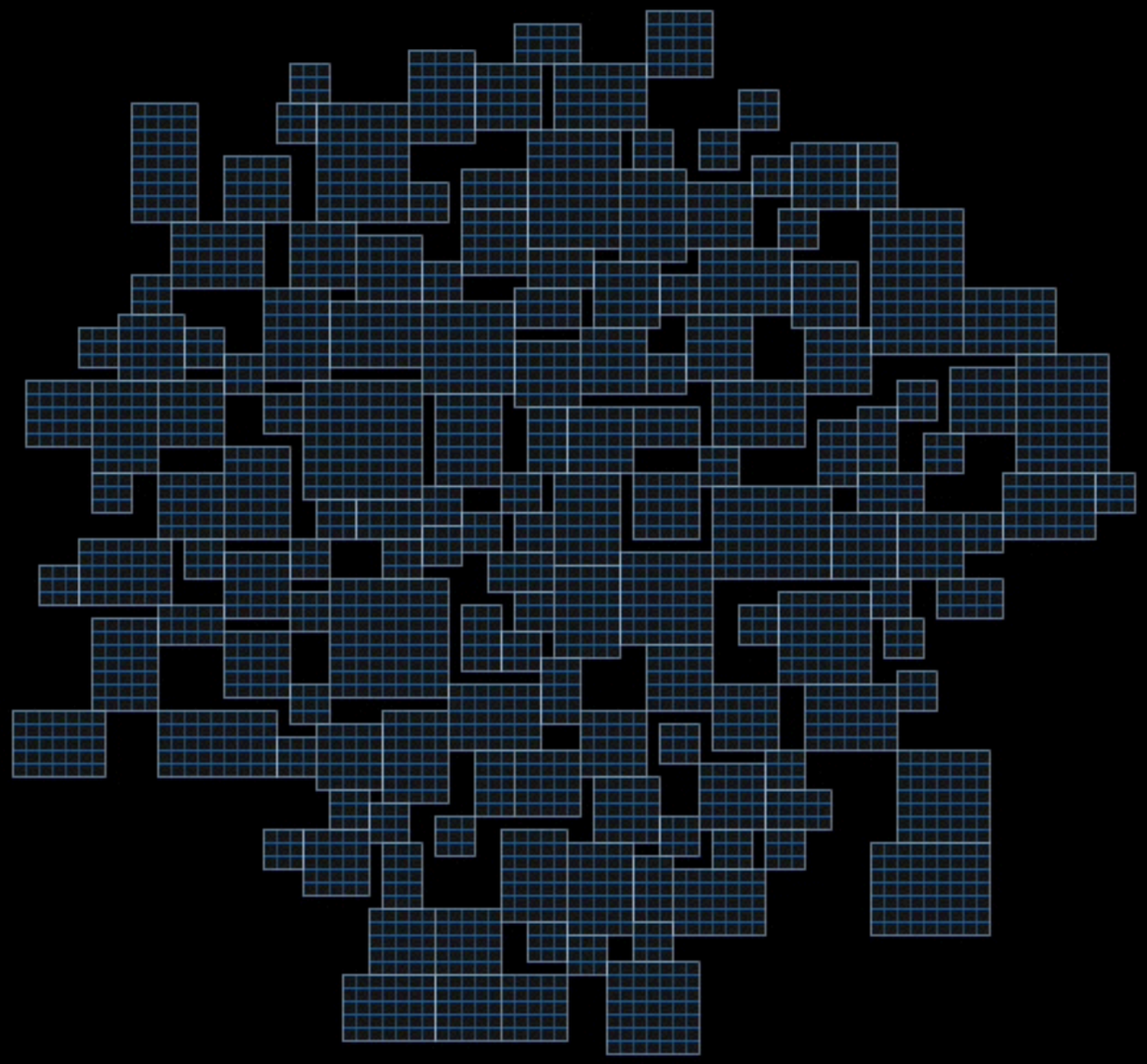


N randomly sized and placed rectangles within radius R.

Prefer small rectangles and limit aspect ratio.

TinyKeep

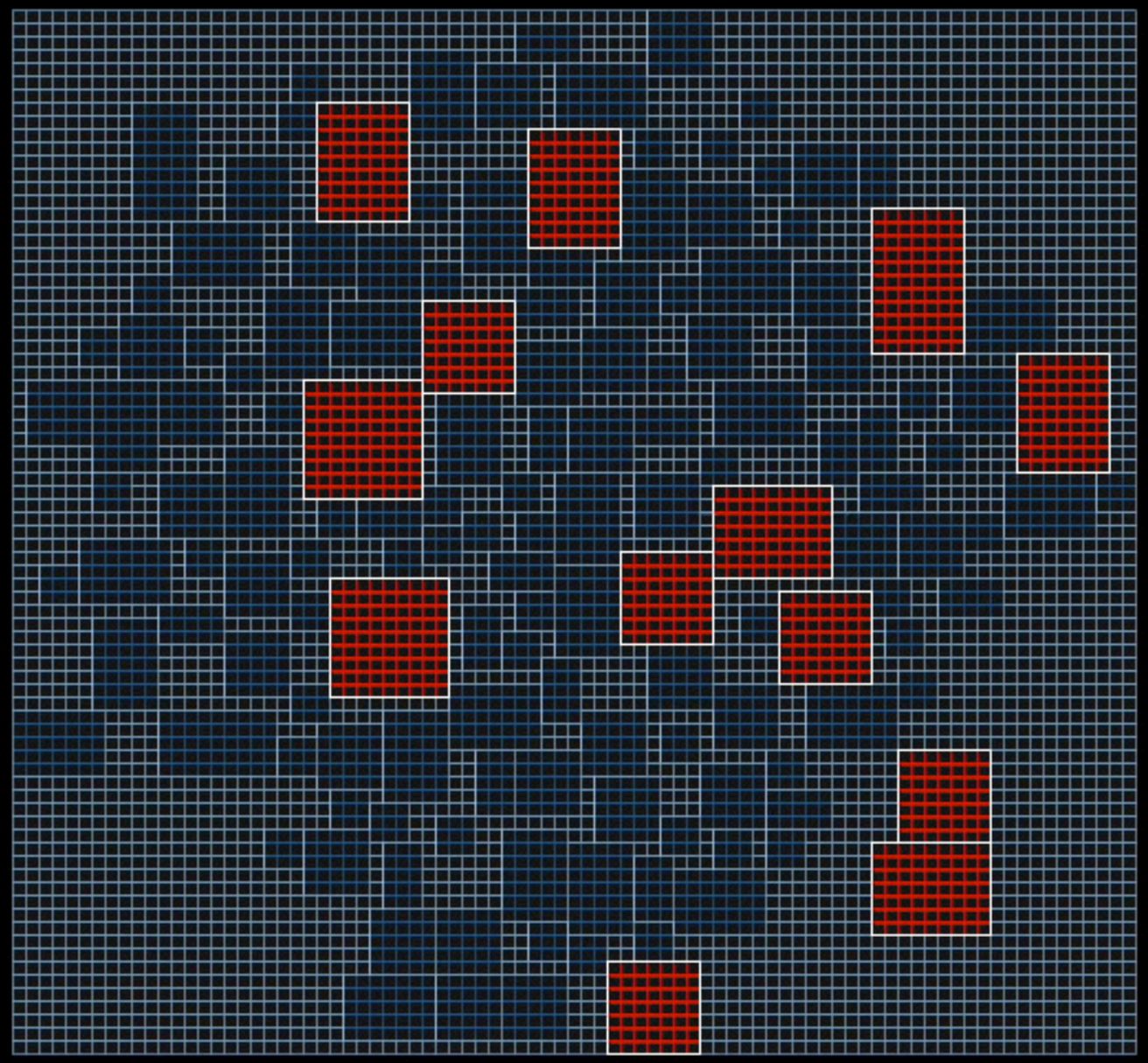
Dungeon Generator



Overlaps produce separation steering force proportional to area.

TinyKeep

Dungeon Generator

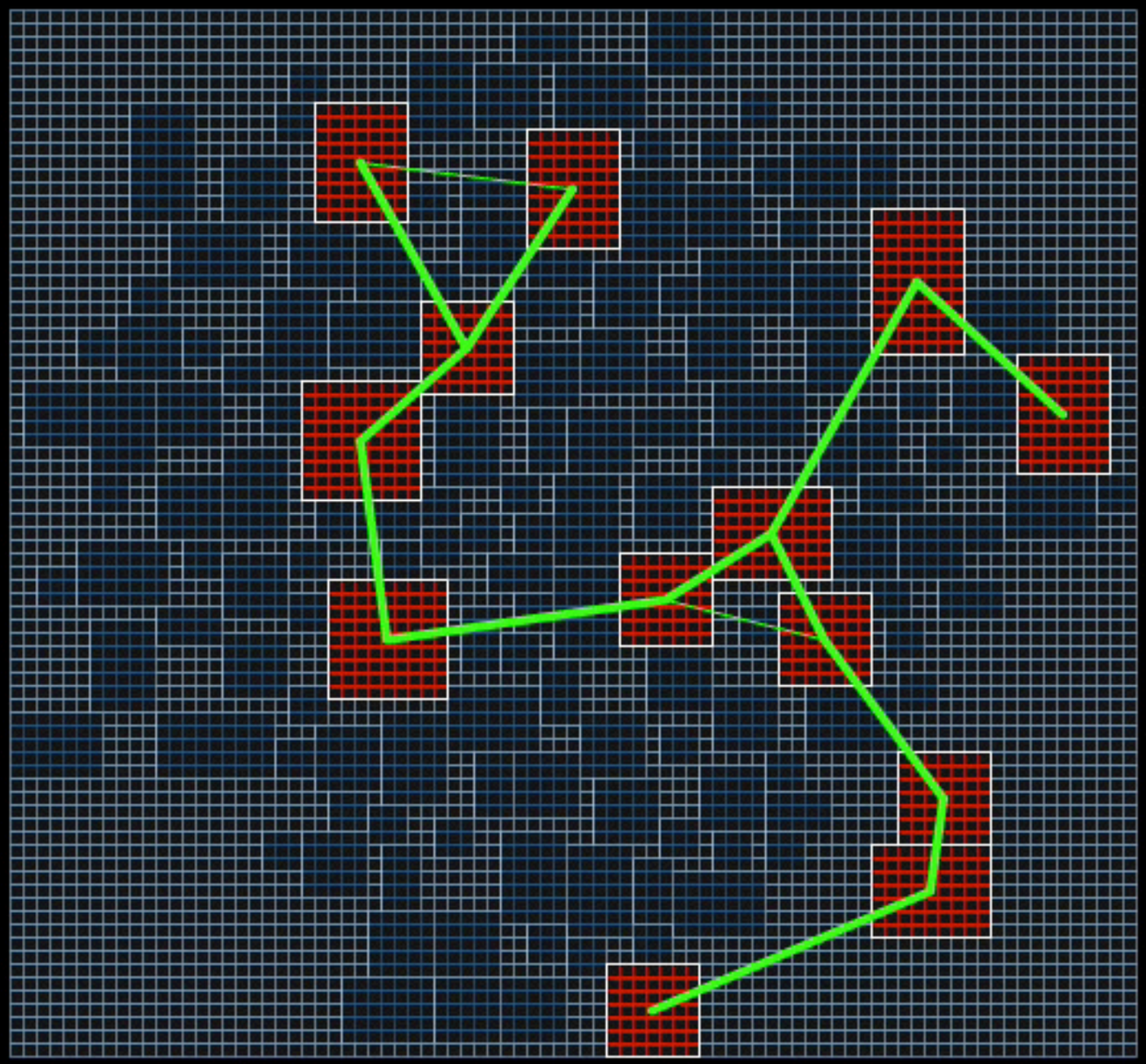


Fill in gaps with
1x1 squares.

Large
rectangles
become rooms.

TinyKeep

Dungeon Generator

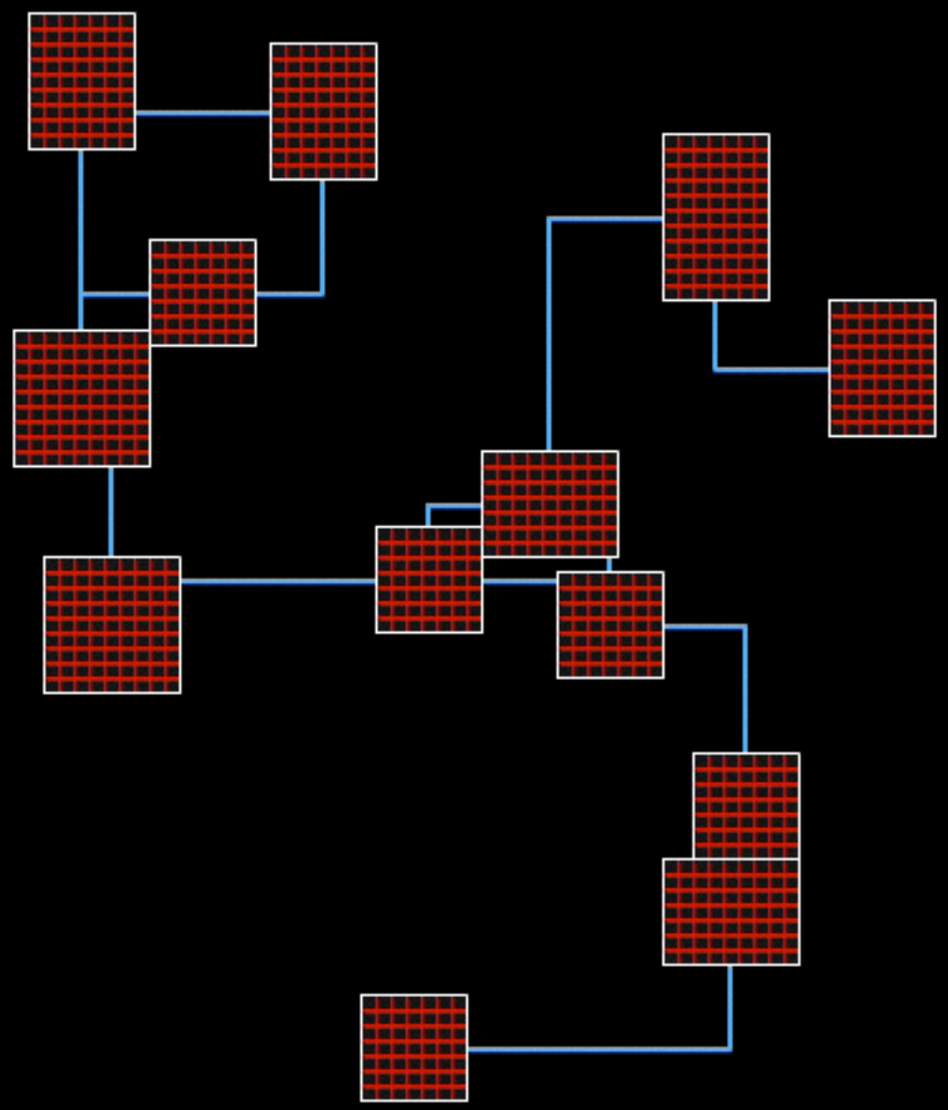


Connect room centres with Delaunay triangulation.

Take min. spanning subgraph + 15% random edges.

TinyKeep

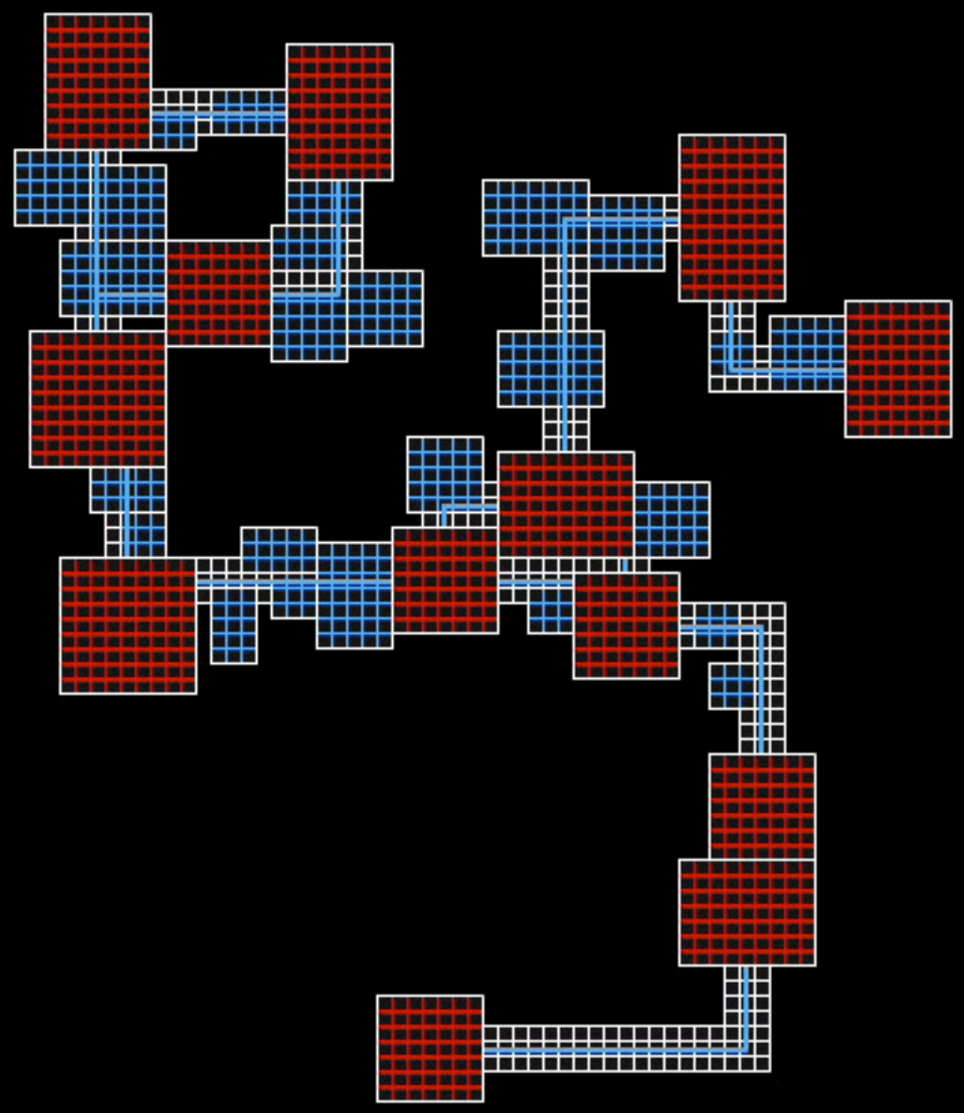
Dungeon Generator



Replace edges with straight connections.

TinyKeep

Dungeon Generator



Any rectangles that overlap become corridors.

Search-Based PCG

- Design = a **search** for good content
- Start with some example content, e.g. a map
- Generate new content by **editing** old content
- Evaluate new examples - are they good?
- Stop when a "good enough" example is found

Search-Based PCG

Content Encodings

- To implement search-based PCG, game content **C** must be **encoded** as a data structure **E**
  - e.g. lists of numbers (parameters) or trees
- A **rendering function** r: E → C generates content from encodings
- An **evaluation function** v: C → [0,1] estimates the quality of content

Search-Based PCG

Editing Content

- We can **change the content** by manipulating the underlying encoding
- A **content edit** is any bit of code which changes the encoding
- PCG is now a **search** for a **sequence of edits** leading to high-quality content
- Similar to pathfinding: *a search for a sequence of moves leading to target location*

PCG as Graph Search

PCG as Graph Search

- Graph
- Node
- Edge
- Start Node
- End Node
- Node Sequence

PCG as Graph Search

Graph	Pathfinding
Node	Location
Edge	Move
Start Node	Start Location
End Node	Target Location
Node Sequence	Path

PCG as Graph Search

Graph	Pathfinding	PCG Search
Node	Location	Content
Edge	Move	Edit
Start Node	Start Location	Initial Content
End Node	Target Location	Final Content
Node Sequence	Path	Design History

PCG as Graph Search

Graph	Pathfinding	PCG Search
Node	Location	Content
Edge	Move	Edit
Start Node	Start Location	Initial Content
End Node	Target Location	Final Content
Node Sequence	Path	Design History

Search for node sequence **vs** search for node

PCG Search

- Search-based PCG requires:
  - Content encoding (defines node)
  - Content edits (defines edges)
  - Content function (renders encodings)
  - Evaluation function (defines quality)
  - Search algorithm (finds high quality node)

Evolutionary Search

- Evolutionary search is a popular approach:
  - Content encoding: **genotype***
  - Content edits: **mutation** and **crossover**
  - Content function generates **phenotype**
  - Evaluation performed by a **fitness** function
  - Search performed by **genetic*** algorithm (GA)

*strictly speaking, there are non-genetic evolutionary algorithms

A Genetic Algorithm

- Genotype (encoding) → Phenotype (content)
- Generate initial population of genotypes
- Copy genotypes to next generation
    - Fitness function **biases** copying
    - Mutation/crossover introduces **variation**
- Terminate when fitness > threshold or converges

Selecting Parents

- Elitism (*the best are copied directly*)
- Truncation (*only the best breed*)
- Fitness proportionate (*prefer fit individuals*)
- Tournament (*run λ small competitions*)

One Point Crossover

[1,17,8,9,10,5,6] [7,9,10,17,1,3,8]

One Point Crossover

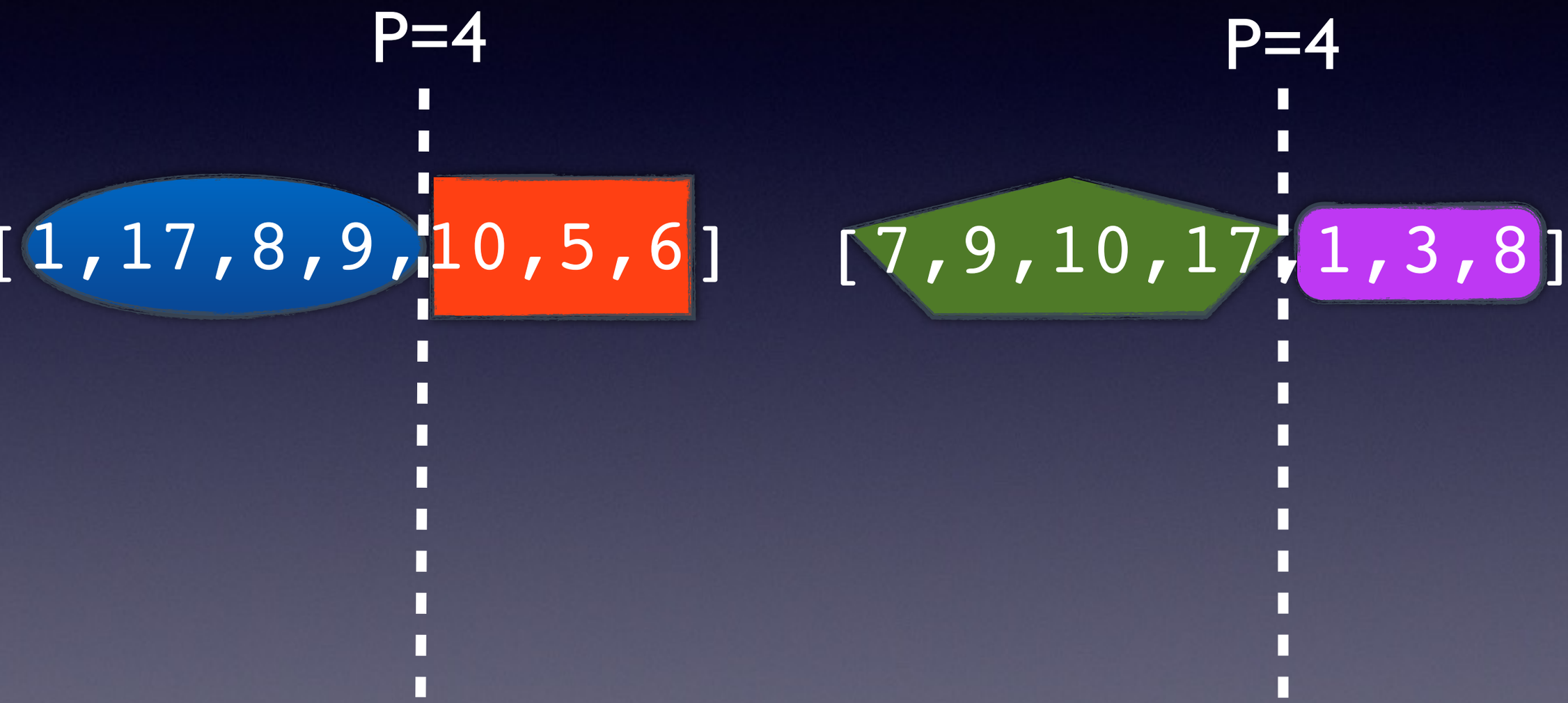
P=4

[1,17,8,9,10,5,6]

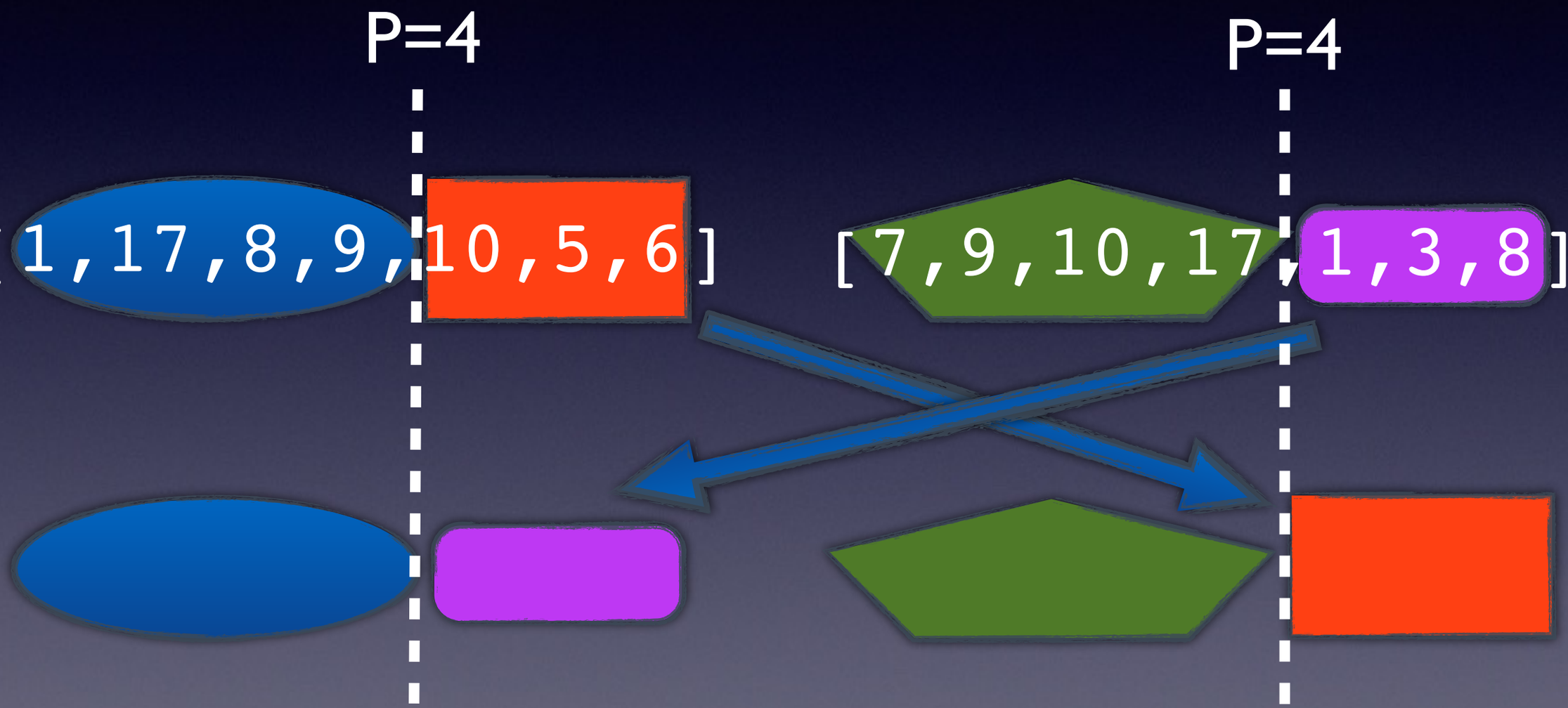
P=4

[7,9,10,17,1,3,8]

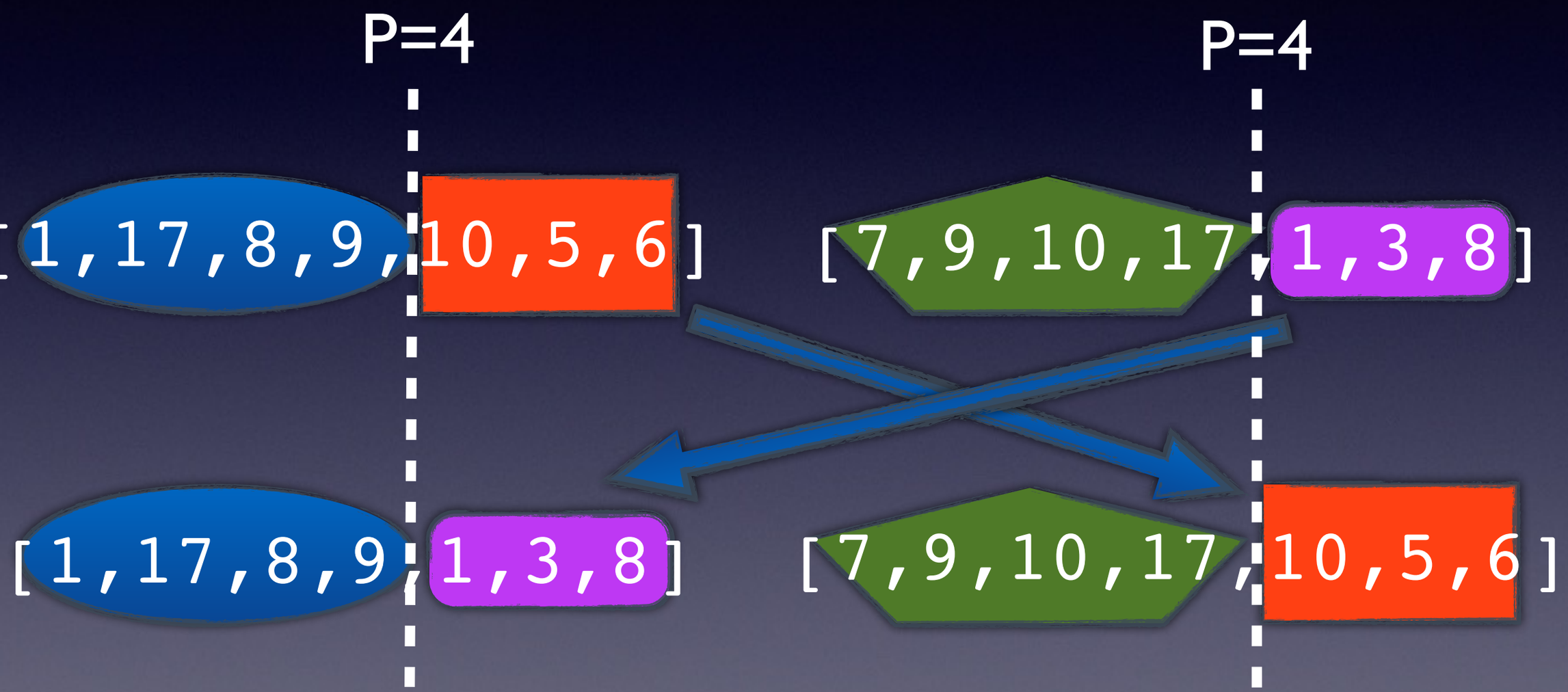
One Point Crossover



One Point Crossover



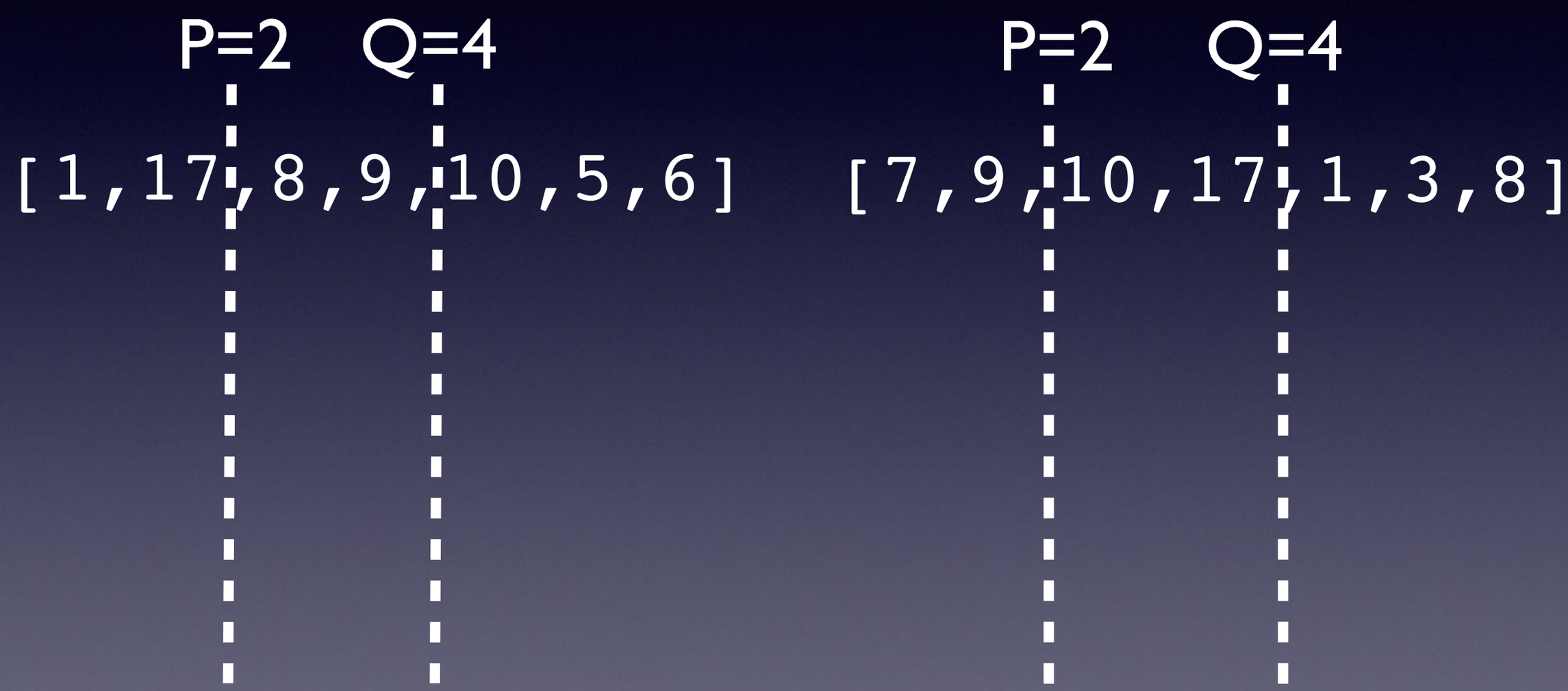
One Point Crossover



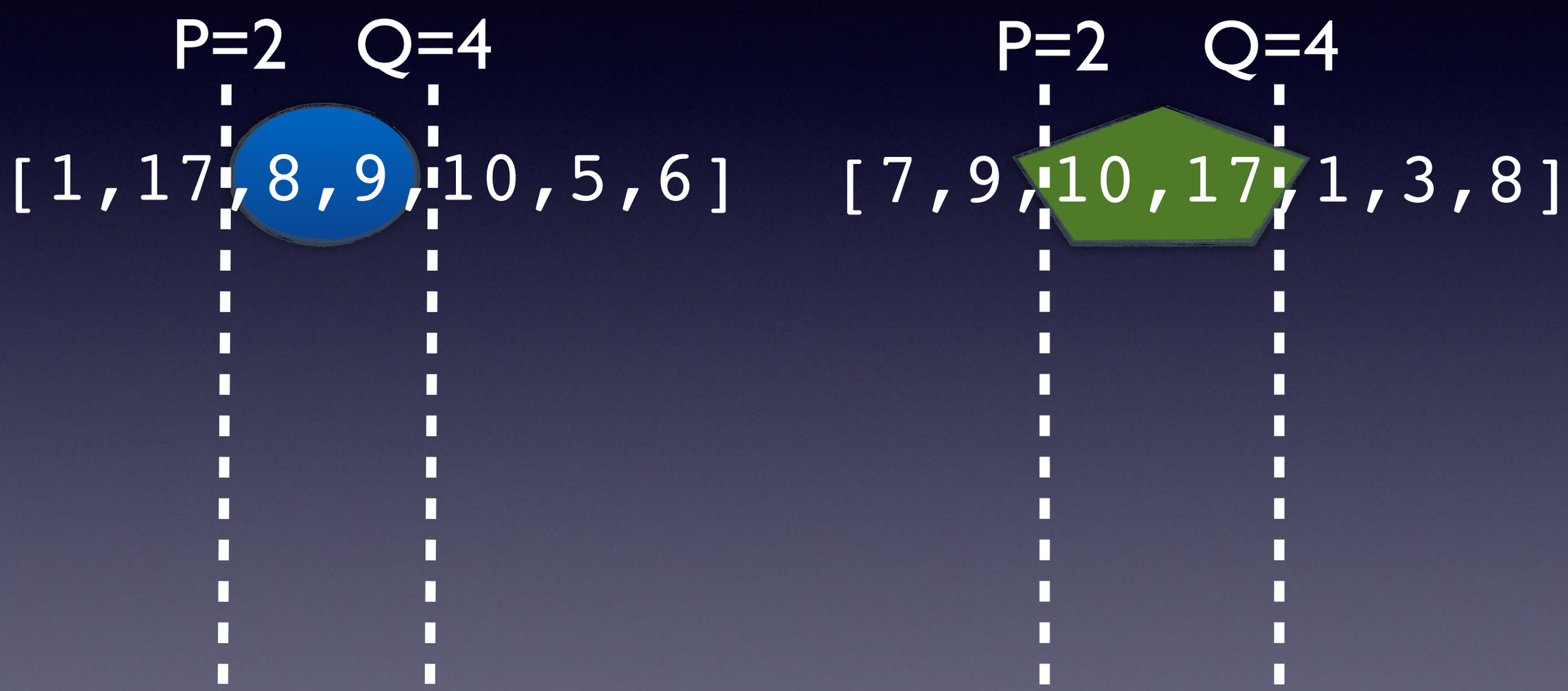
Two Point Crossover

[1,17,8,9,10,5,6] [7,9,10,17,1,3,8]

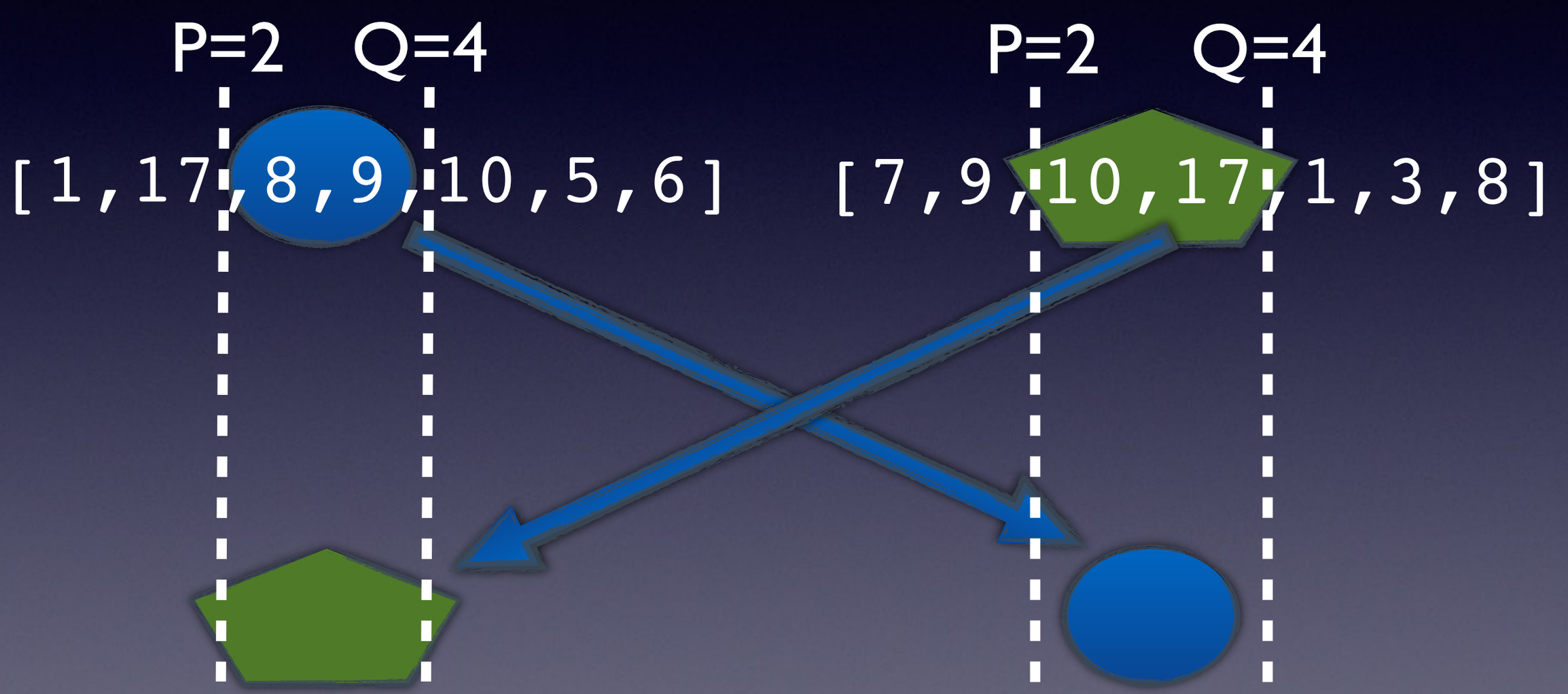
Two Point Crossover



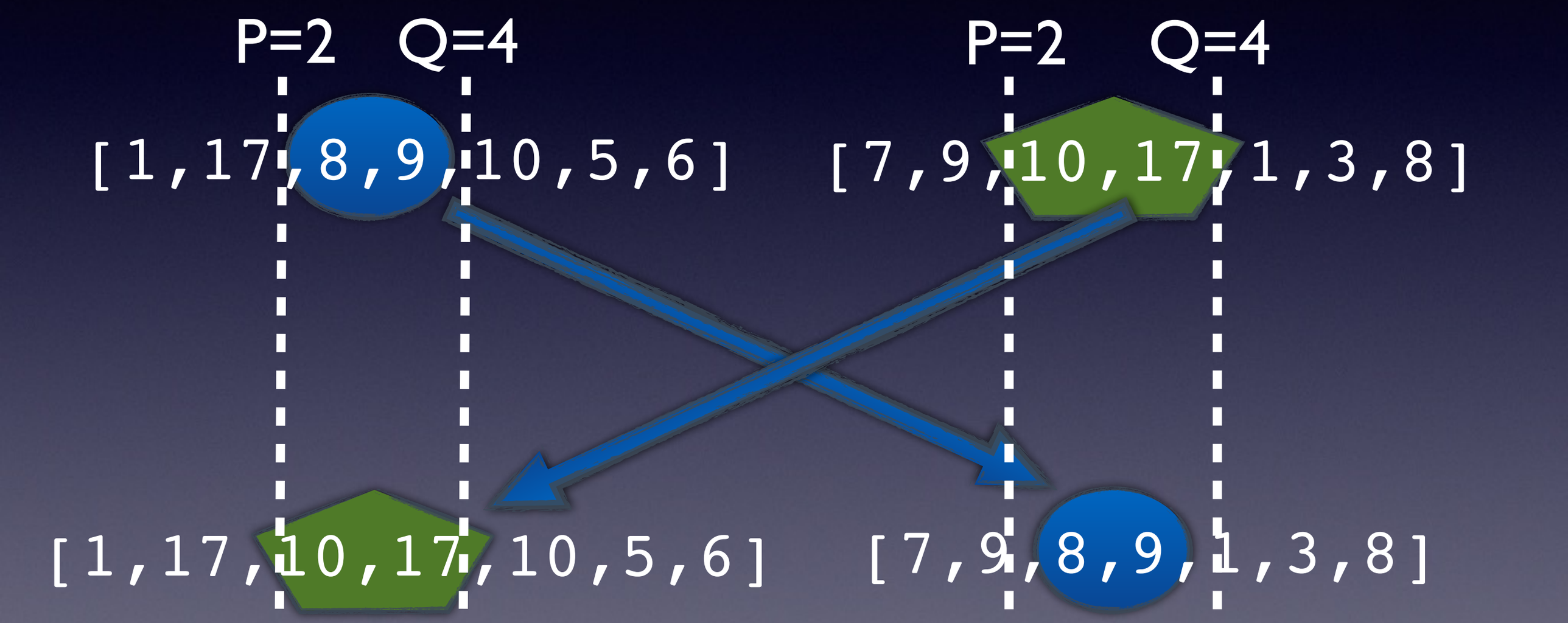
Two Point Crossover



Two Point Crossover

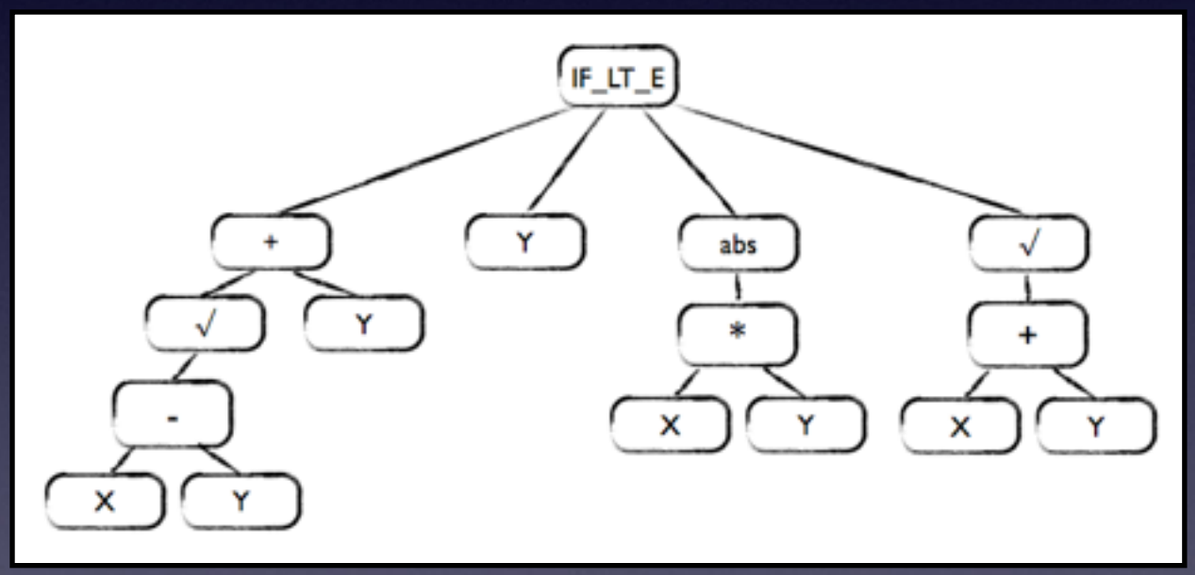


Two Point Crossover

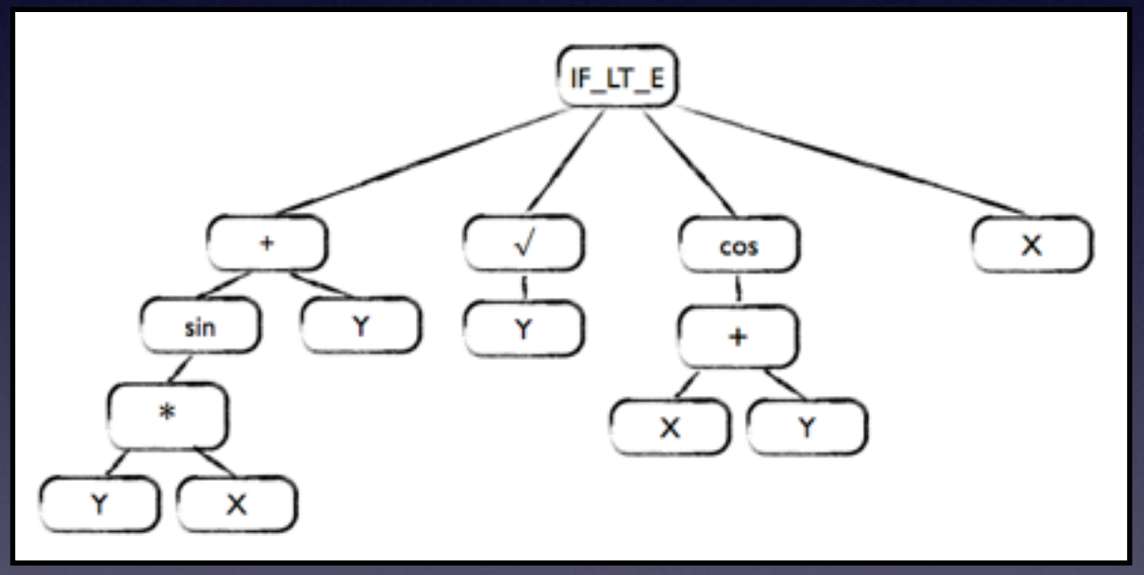


Tree Crossover

Parent A

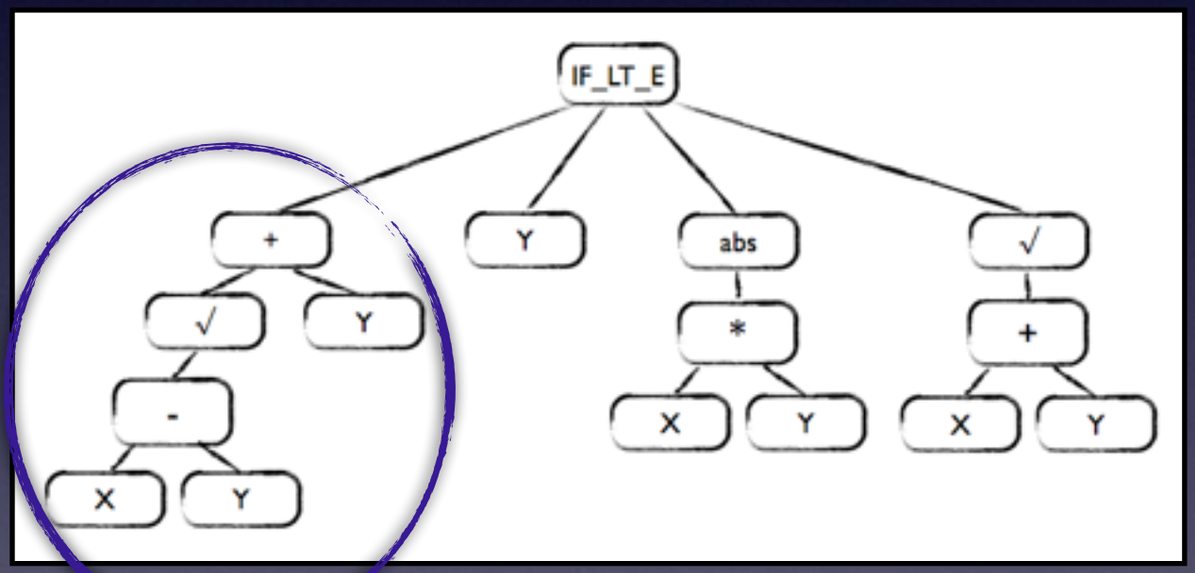


Parent B

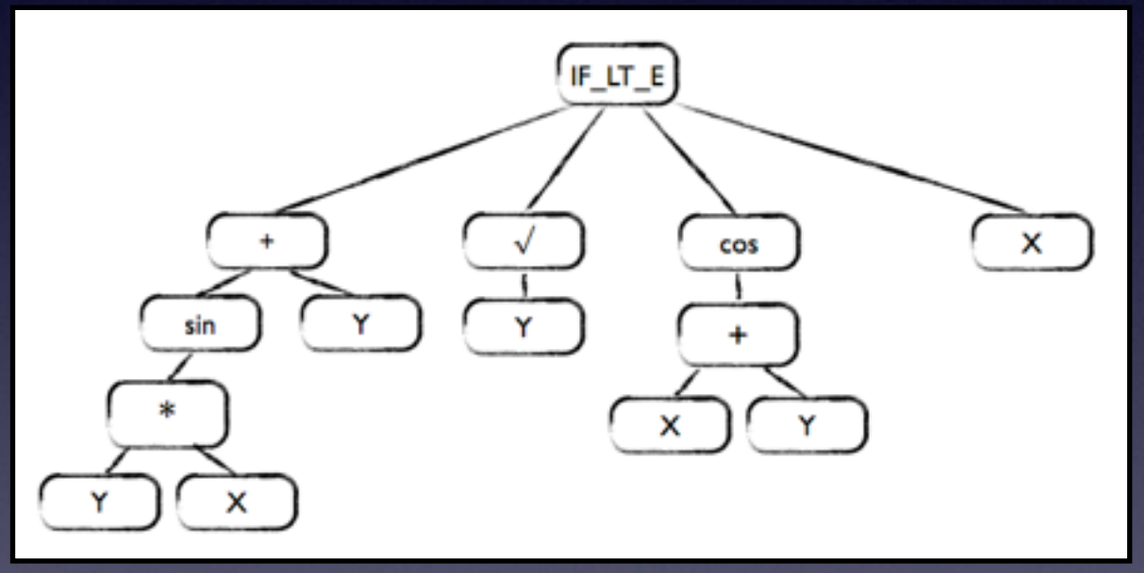


Tree Crossover

Parent A



Parent B

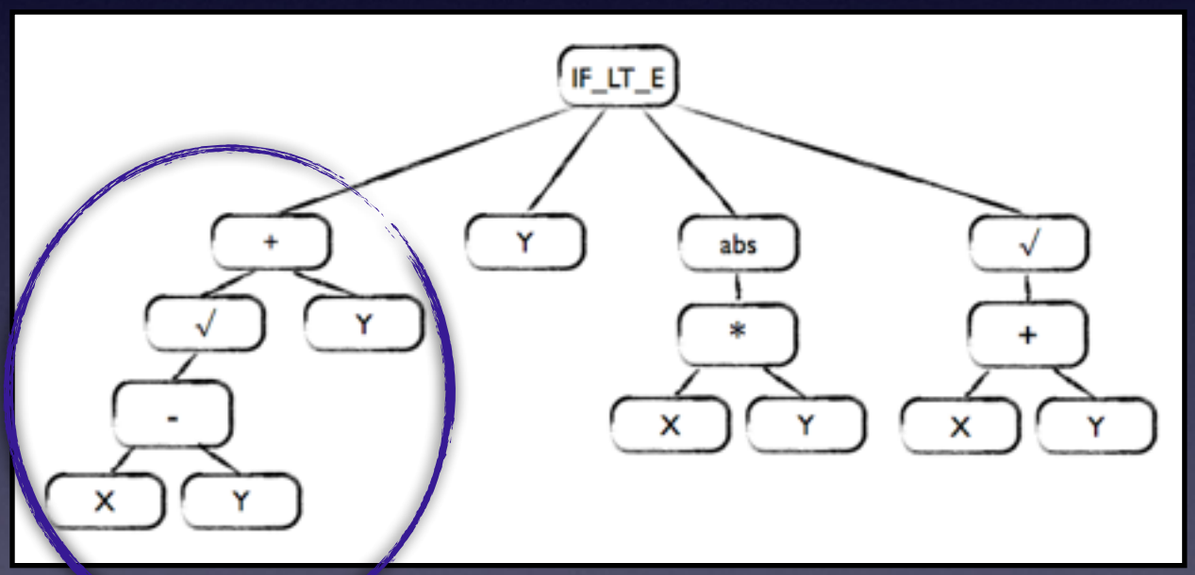


Subtree of A

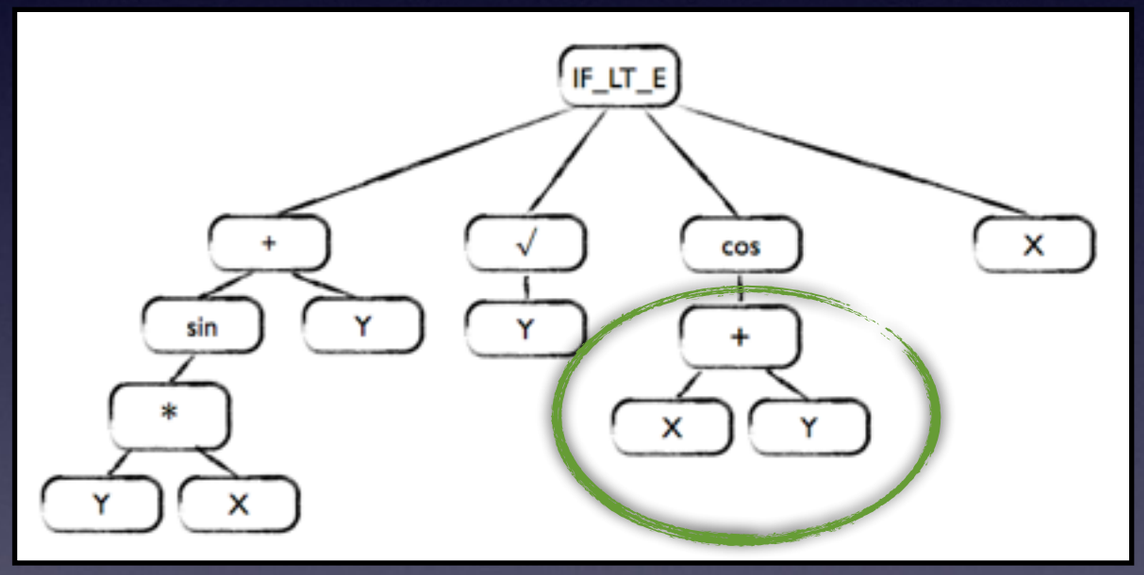
$\sqrt{(X-Y)}+Y$

Tree Crossover

Parent A



Parent B



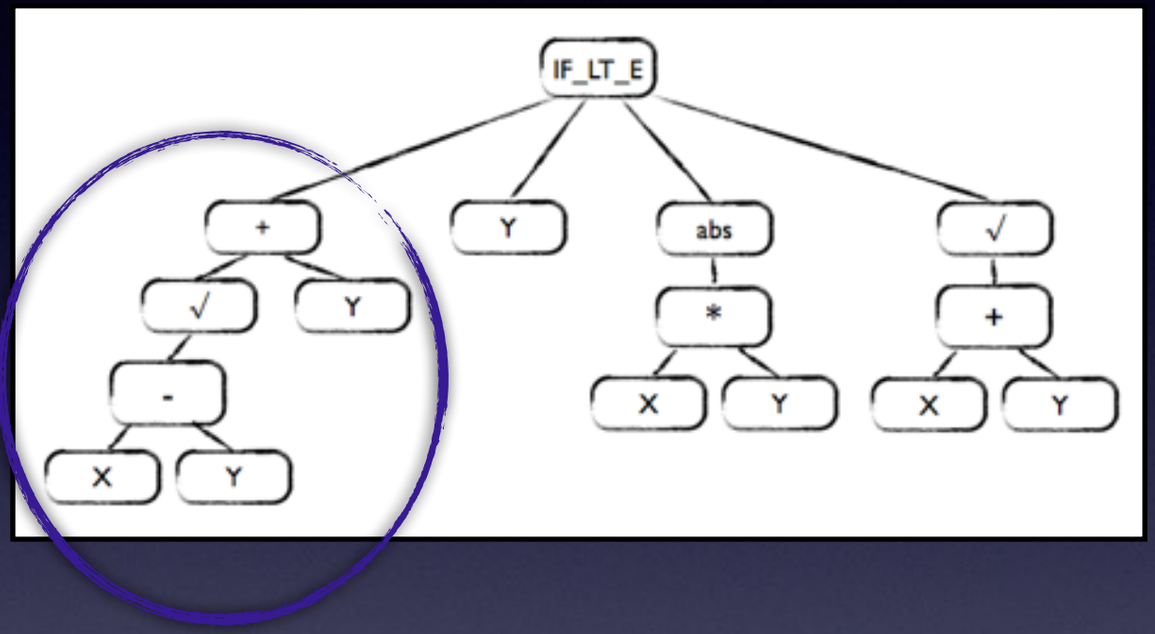
Subtree of A

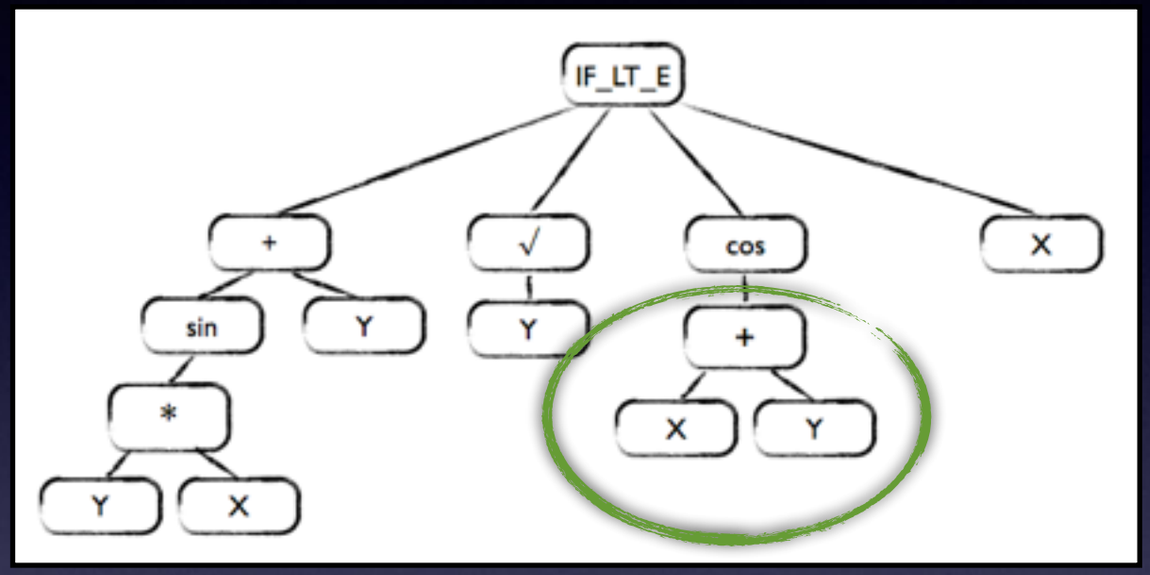
$\sqrt{(X-Y)}+Y$

Subtree of B

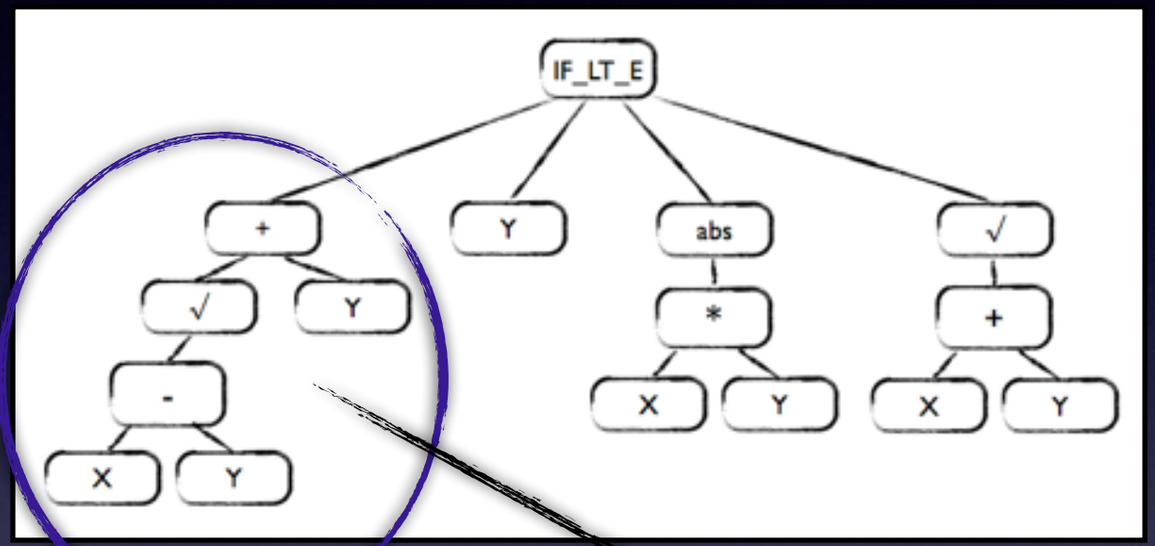
X+Y

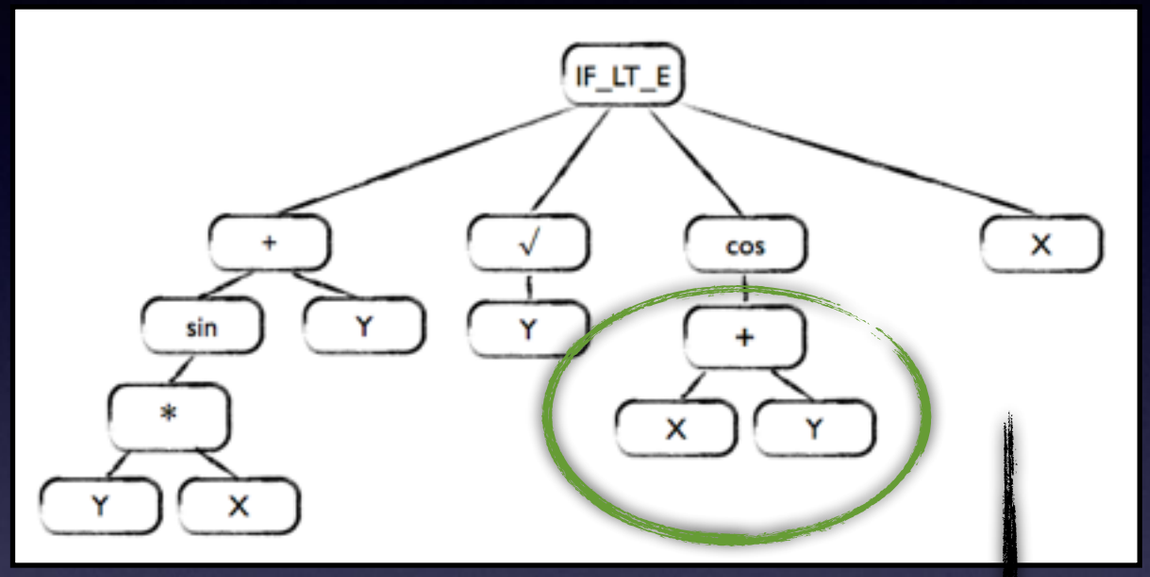
Tree Crossover



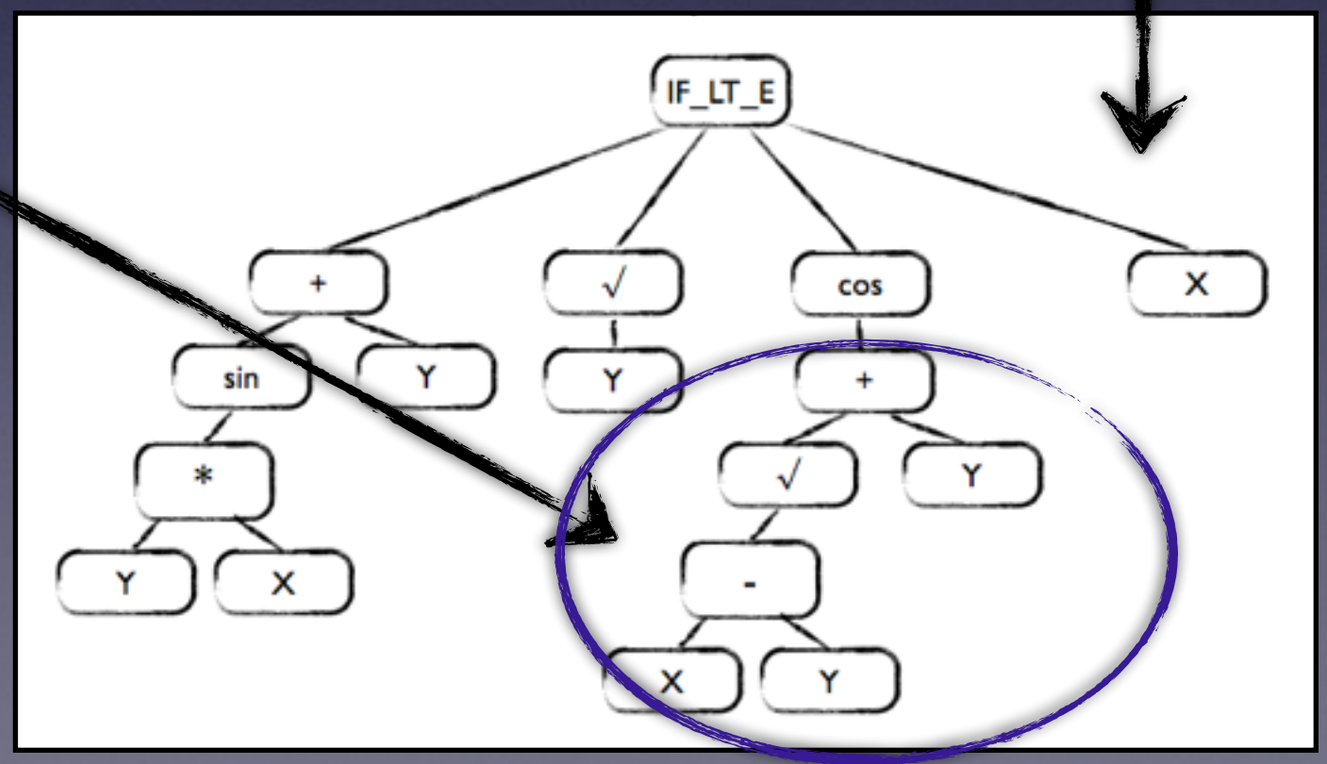


Tree Crossover

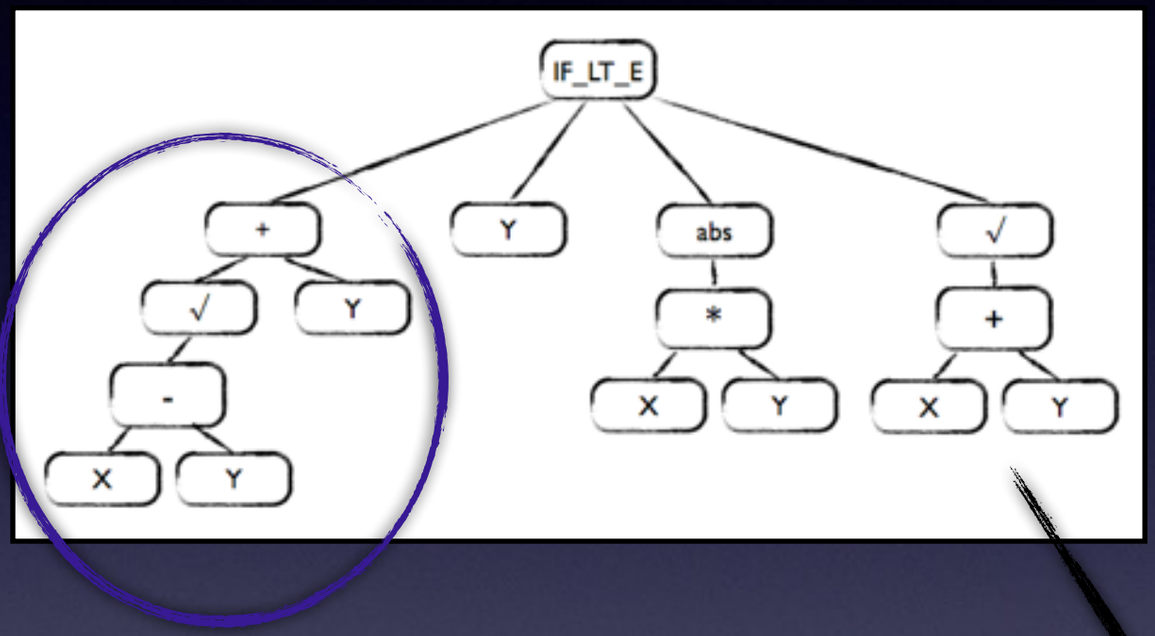


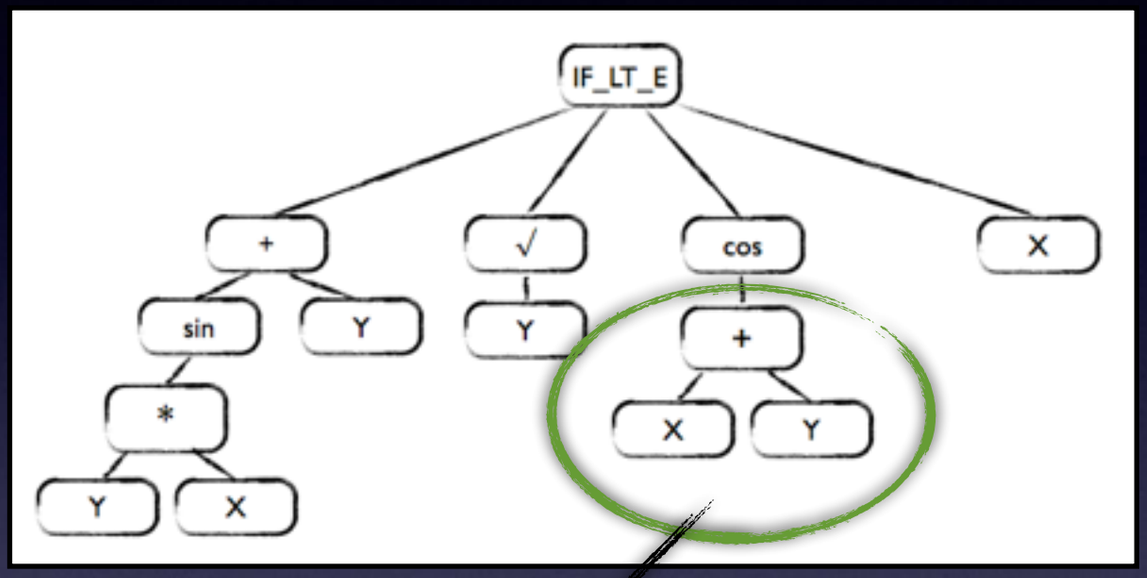


Child 1
= Subtree of A
inserted into tree B

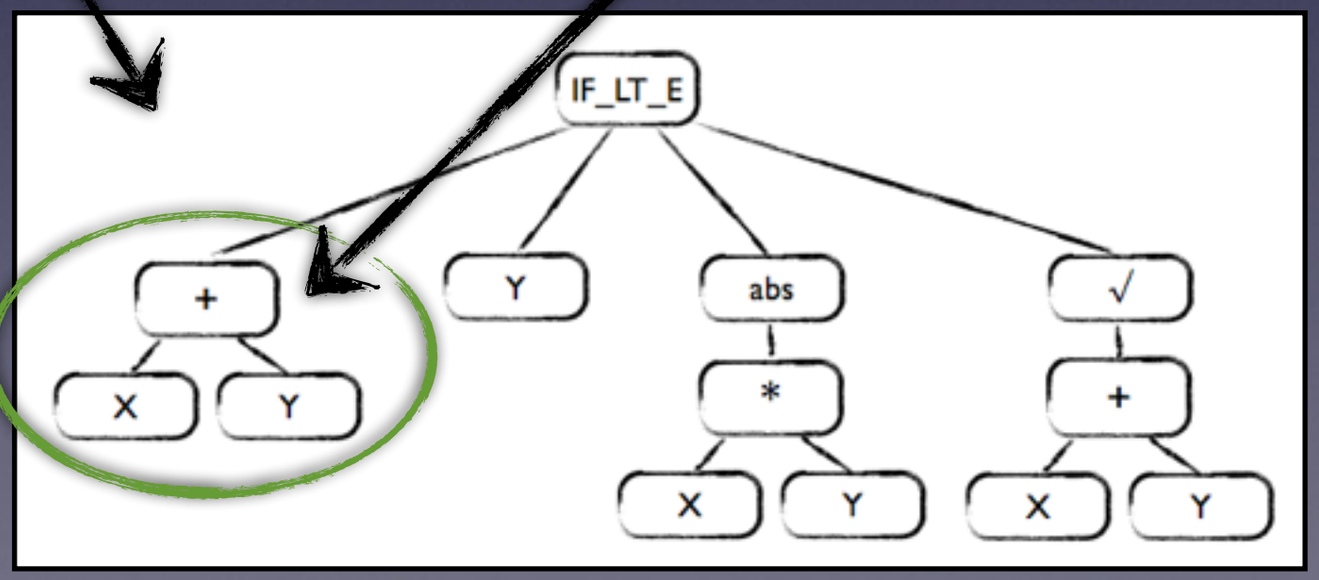


Tree Crossover





Child 2
= Subtree of B
inserted into tree A

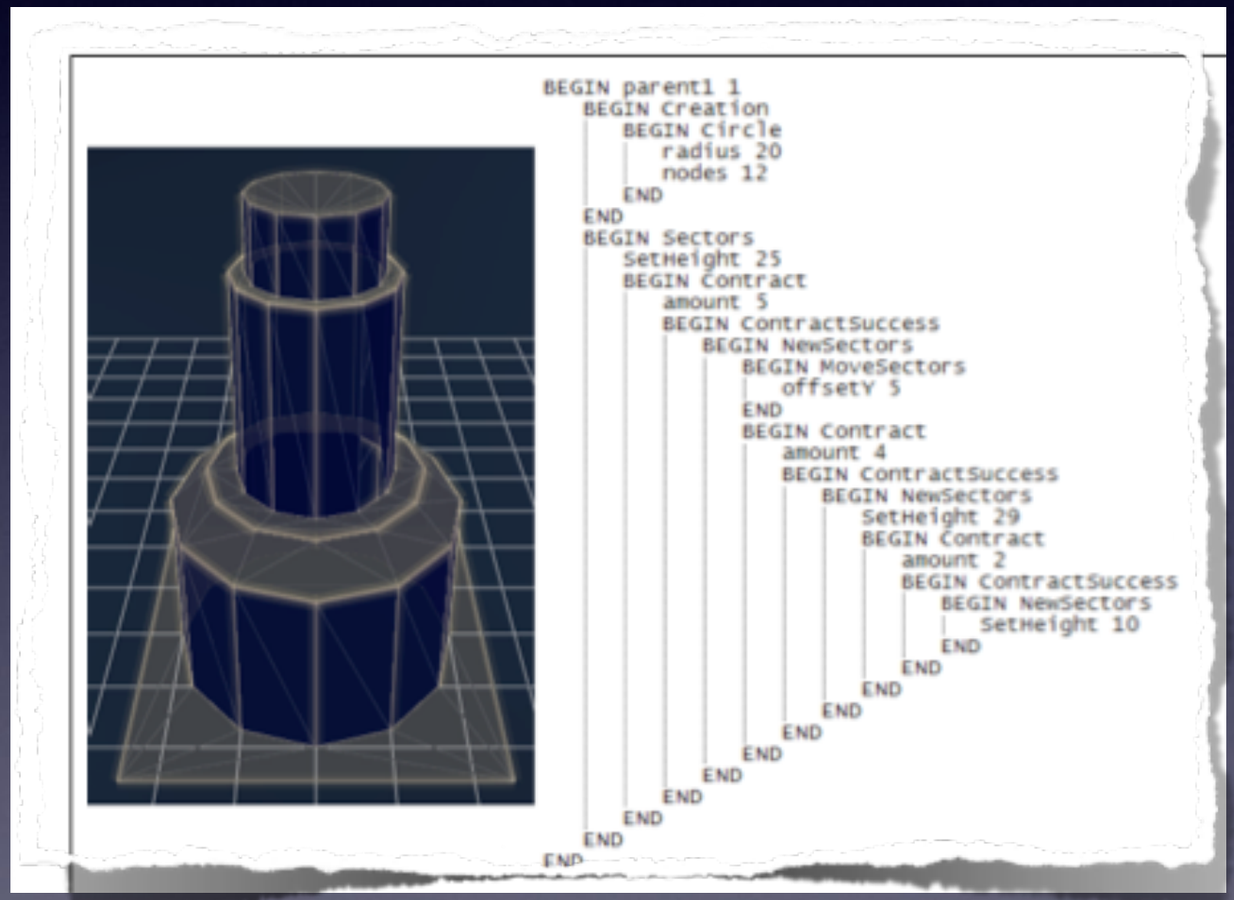


Fitness Functions

- **Direct**: measure static properties of the phenotype content.
- **Simulated**: measure aspects of simulated play with content.
- **Interactive**: explicit or implicit rating by people.

Evolving Buildings

- Bespoke scripts control shape of towers in *Subversion*
- Coded as a script tree
- Crossover and mutation
- Strong and weak forms of both crossover and mutation
    - Sliders control strength of each operator



http://www.doc.ic.ac.uk/~sgc/papers/martin_evogames10.pdf

Evolving Buildings

Further Reading

- Shaker, Togelius & Nelson (2014) *Procedural Content Generation in Games,* chapters 1-3. **Free online** http://pcgbook.com/

- Michael Cook, *The Saturday Papers* blog http://www.gamesbyangelina.org/cat/the-saturday-papers/

- PROCJAM http://itch.io/jam/procjam