ANGELINA versus Terry Cavanaugh

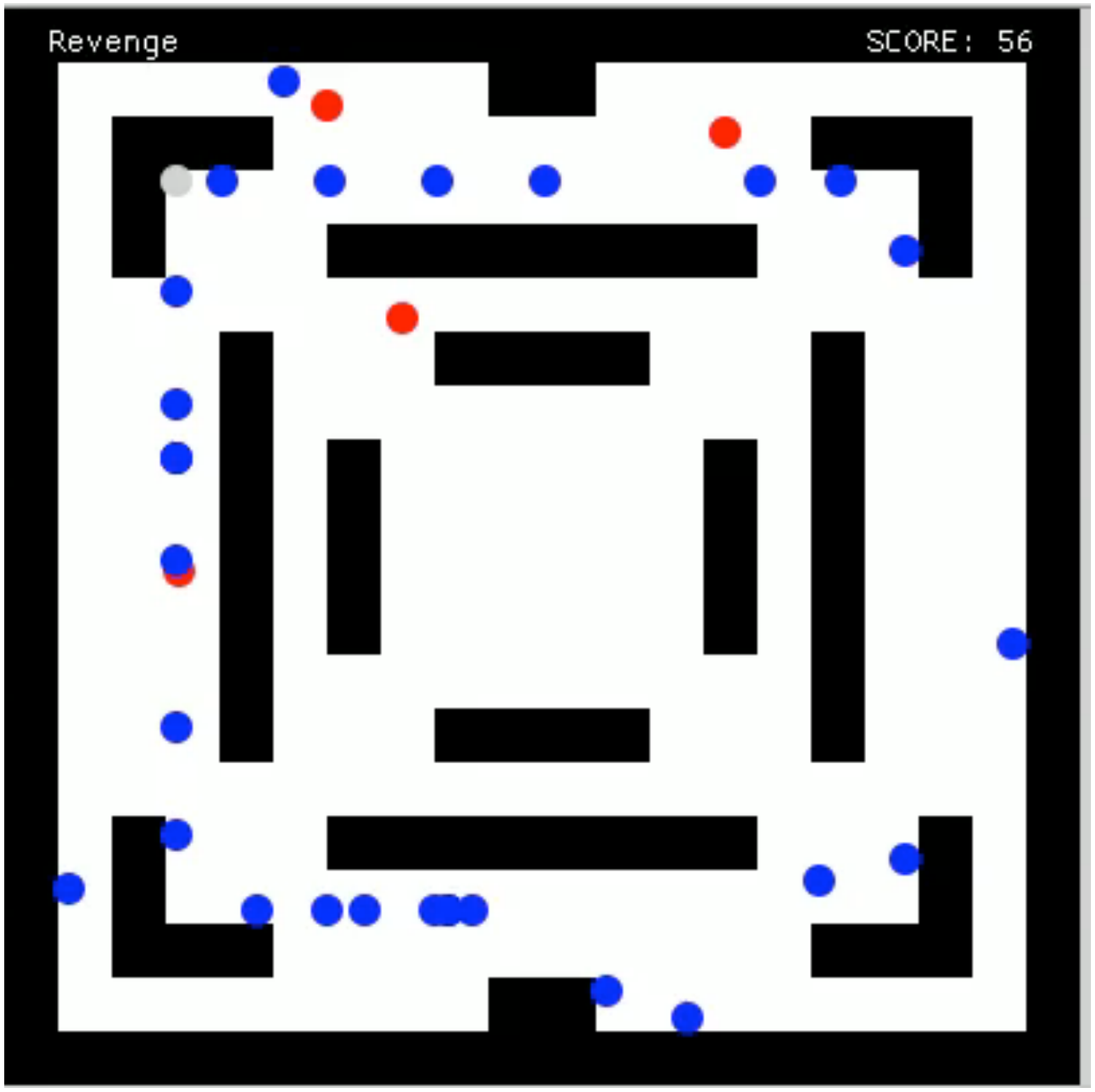
Or "Mechanic Discovery through Simulation"

Michael Cook
Computational Creativity Group
Imperial College, London

www.gamesbyangelina.org

Can you automatically design games?

Arcade Games



Play *Revenge* at www.gamesbyangelina.org/games

Arcade Games

Grammar-based Rule Generation

Arcade Games

Grammar-based Rule Generation

- Define rule templates, e.g.
  <TYPE, TYPE, EFFECT, EFFECT,SCORE>
- Define a set of possibilities, e.g.
  TYPE = Player, Red, Blue, Green

Arcade Games

Grammar-based Rule Generation

- Define rule templates, e.g.
  <TYPE, TYPE, EFFECT, EFFECT,SCORE>
- Define a set of possibilities, e.g.
  TYPE = Player, Red, Blue, Green
- Good ratio of "good rules" :"rules".
- Rationalisation: designer involvement.

Platform Games

Play *The Conservation of Emily* at www.gamesbyangelina.org/games

Platform Games

Simulation-Assisted Parameter Search

Platform Games

Simulation-Assisted Parameter Search

- Define candidate game variables to target, e.g. player.jumpVelocity
- Evolutionary search optimises powerups for e.g. reachability gain, level flow

Platform Games

Simulation-Assisted Parameter Search

- Define candidate game variables to target, e.g. player.jumpVelocity
- Evolutionary search optimises powerups for e.g. reachability gain, level flow
- Reliant yet again on human assistance
- One-time powerup use a bit boring? We can do better!

What Do We Want?

What Do We Want?

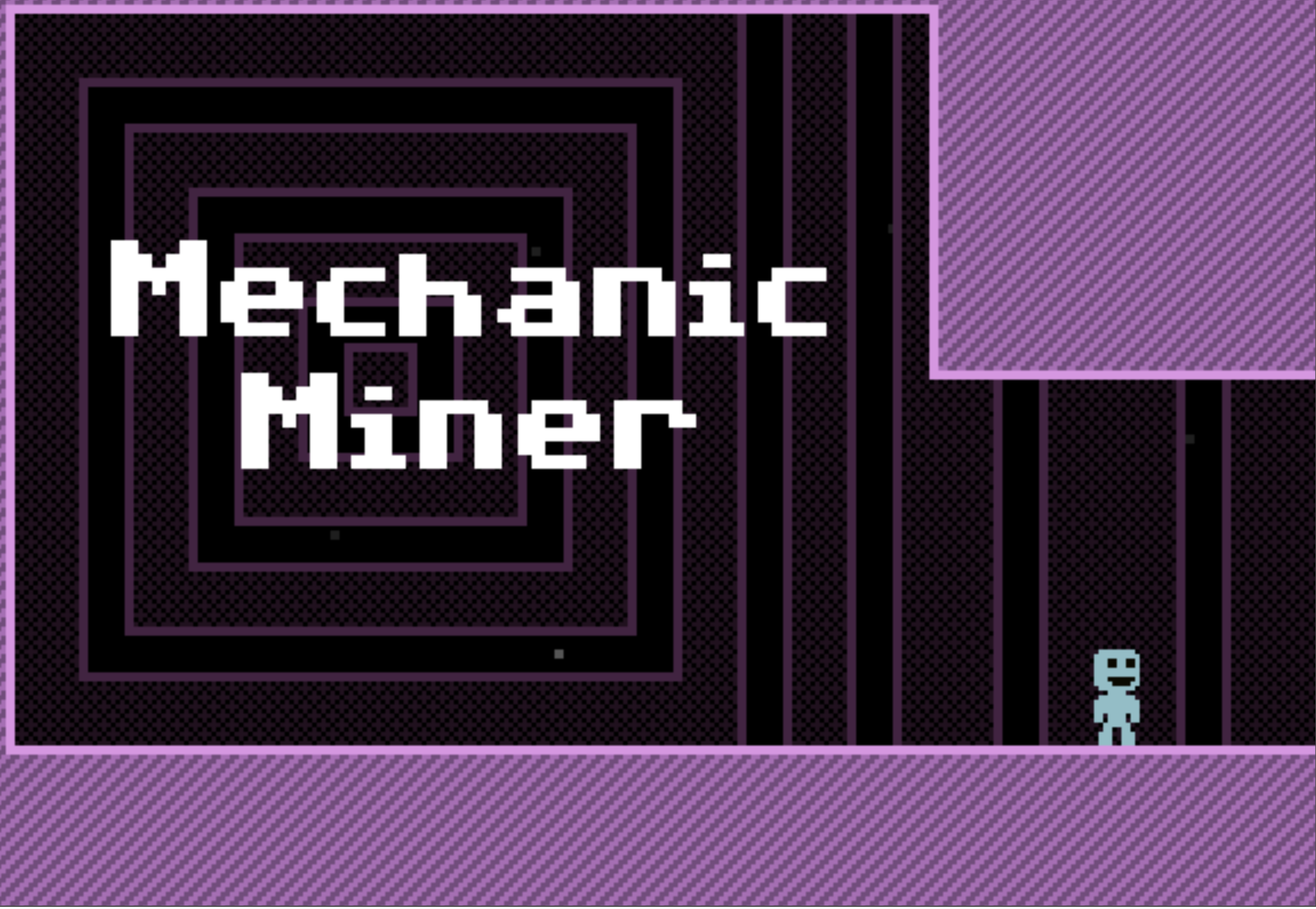
- Reduce reliance on humans

What Do We Want?

- Reduce reliance on humans
- Generate true ‘mechanics’

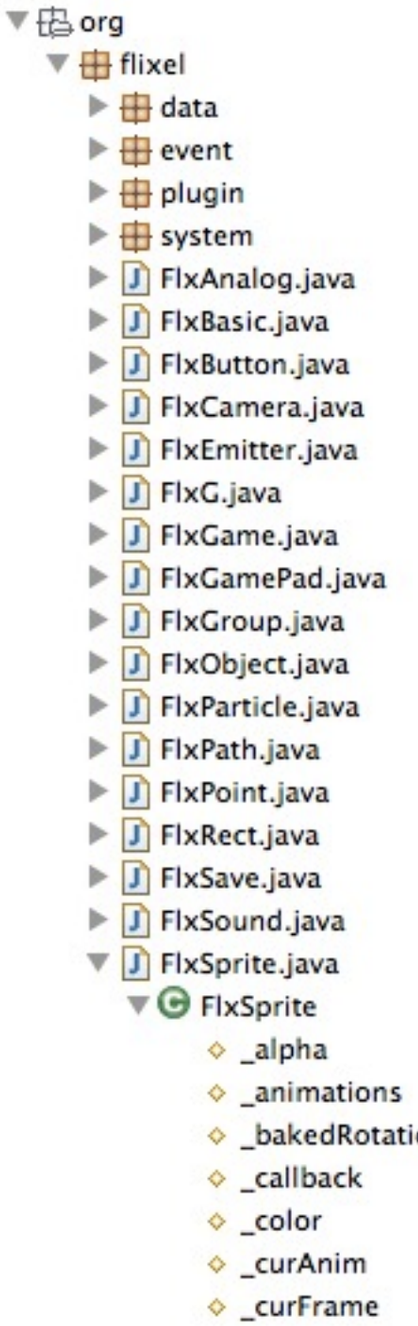
What Do We Want?

- Reduce reliance on humans
- Generate true ‘mechanics’
- Surprise us!


Mechanic
Miner

Mechanic Miner

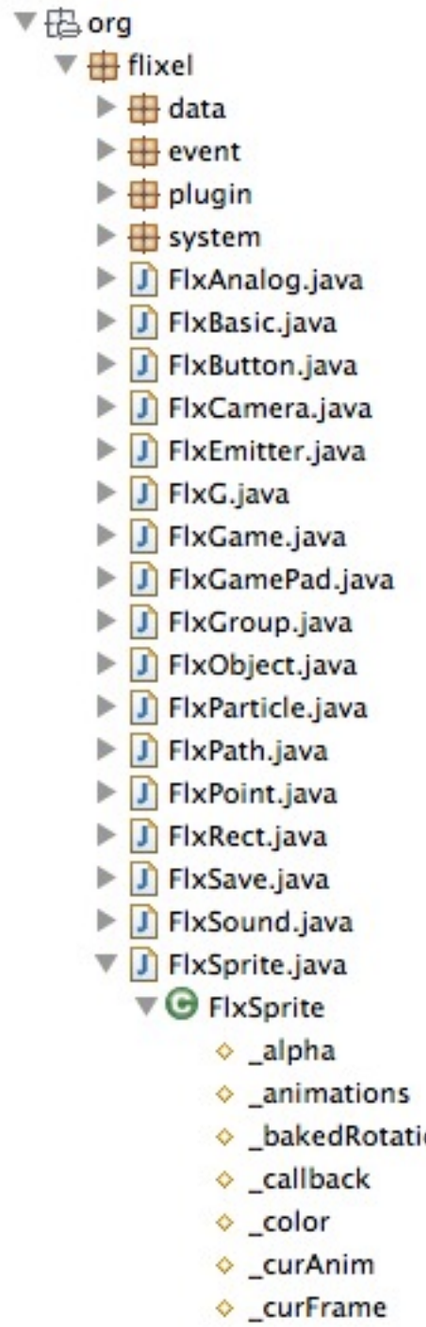
Generation



Mechanic Miner

Generation

- Java Reflection lets us inspect code at runtime and reason about it,
  e.g. FlxSprite.getClass().getFields();

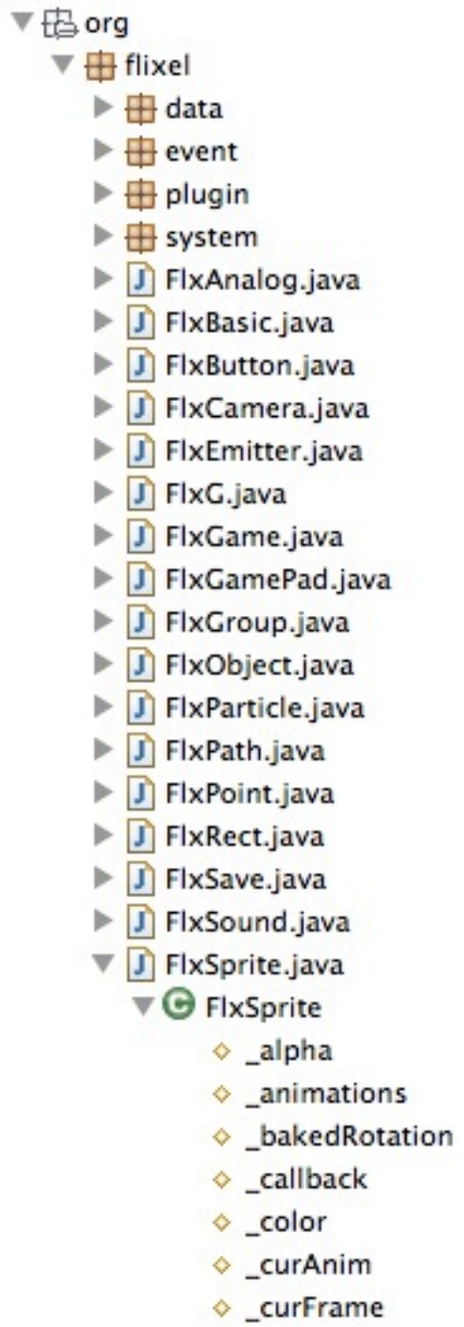


Mechanic Miner

Generation

- Java Reflection lets us inspect code at runtime and reason about it,
  e.g. FlxSprite.getClass().getFields();

- Pick a field, attach a modifier (e.g. HALVE, INVERTSIGN) and you're good to go.



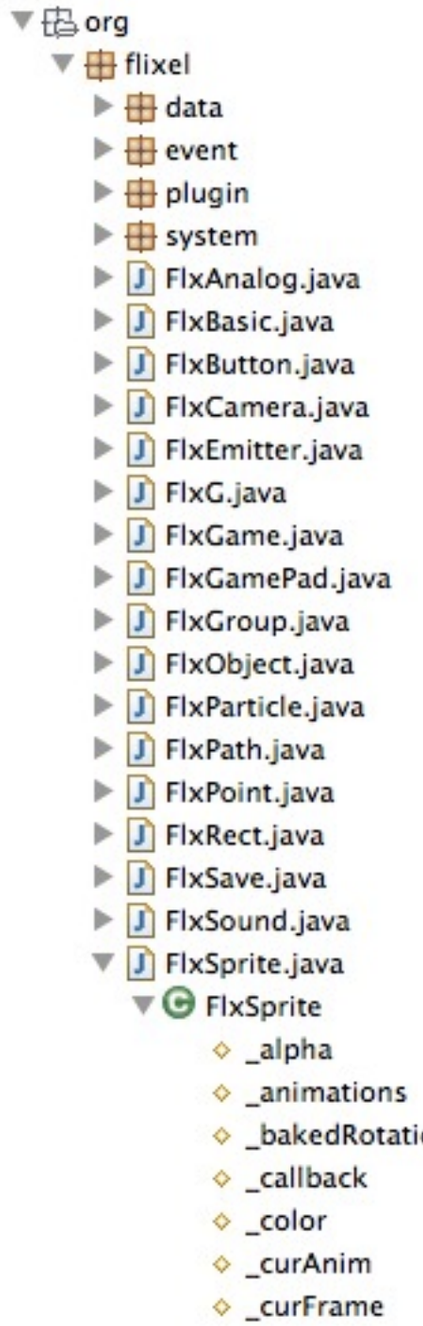
Mechanic Miner

Generation

- Java Reflection lets us inspect code at runtime and reason about it,
  e.g. FlxSprite.getClass().getFields();

- Pick a field, attach a modifier (e.g. HALVE, INVERTSIGN) and you're good to go.

- Can reason about instance and static fields (after a headache)



Mechanic Miner

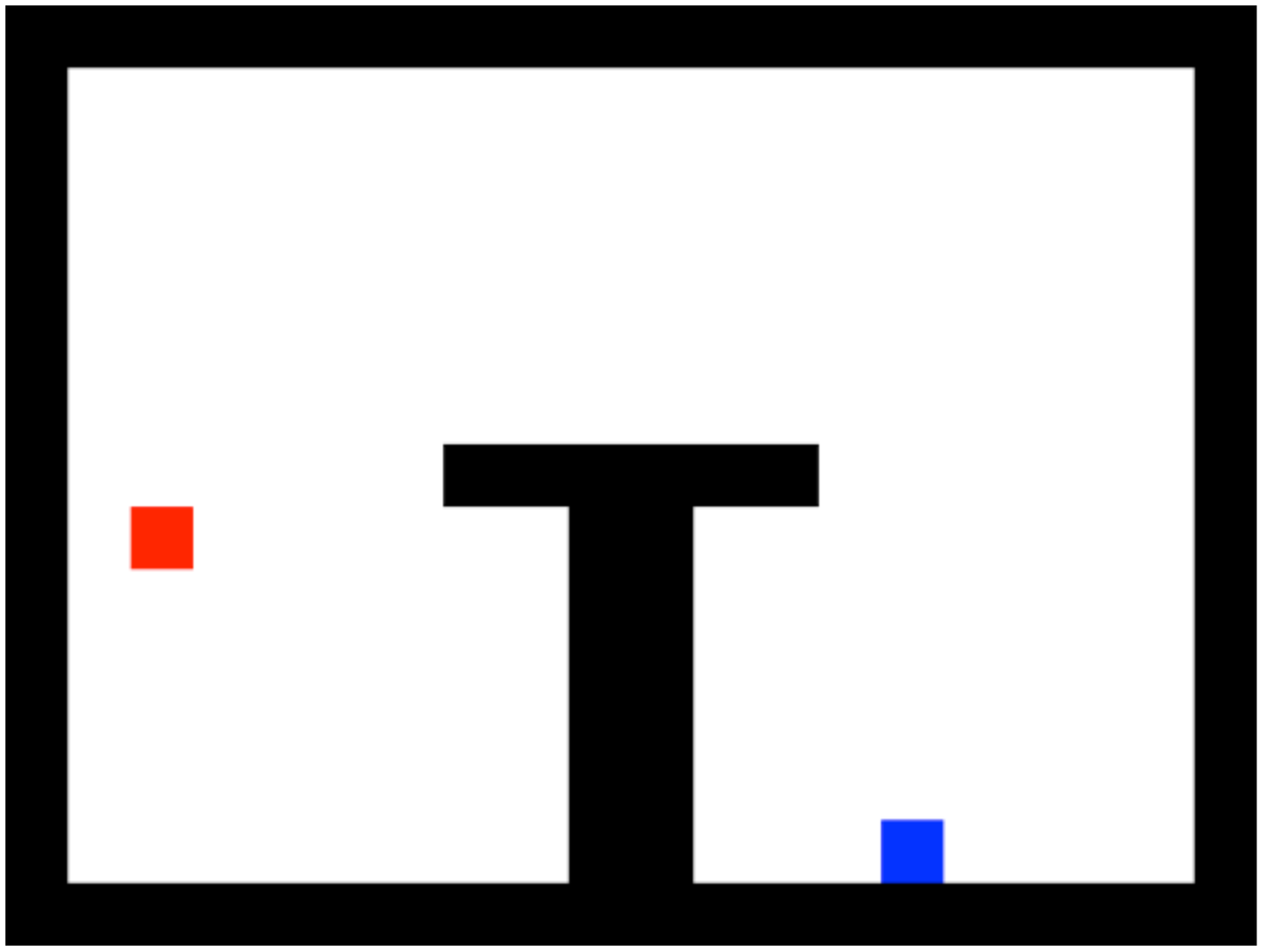
Evaluation

Take an ‘unsolvable’ level, and simulate playing it.

Mechanic Miner

Evaluation

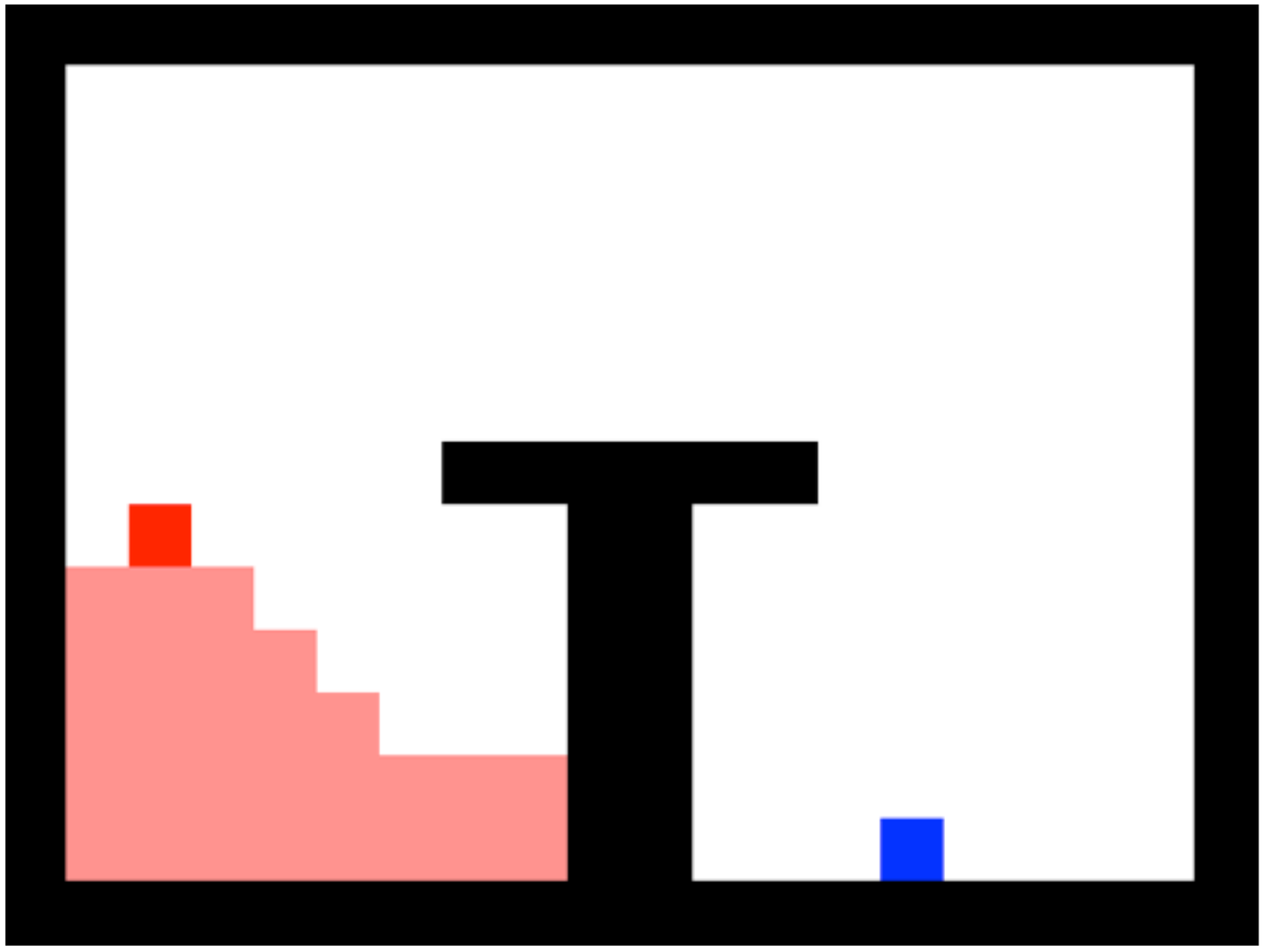
Take an ‘unsolvable’ level, and simulate playing it.



Mechanic Miner

Evaluation

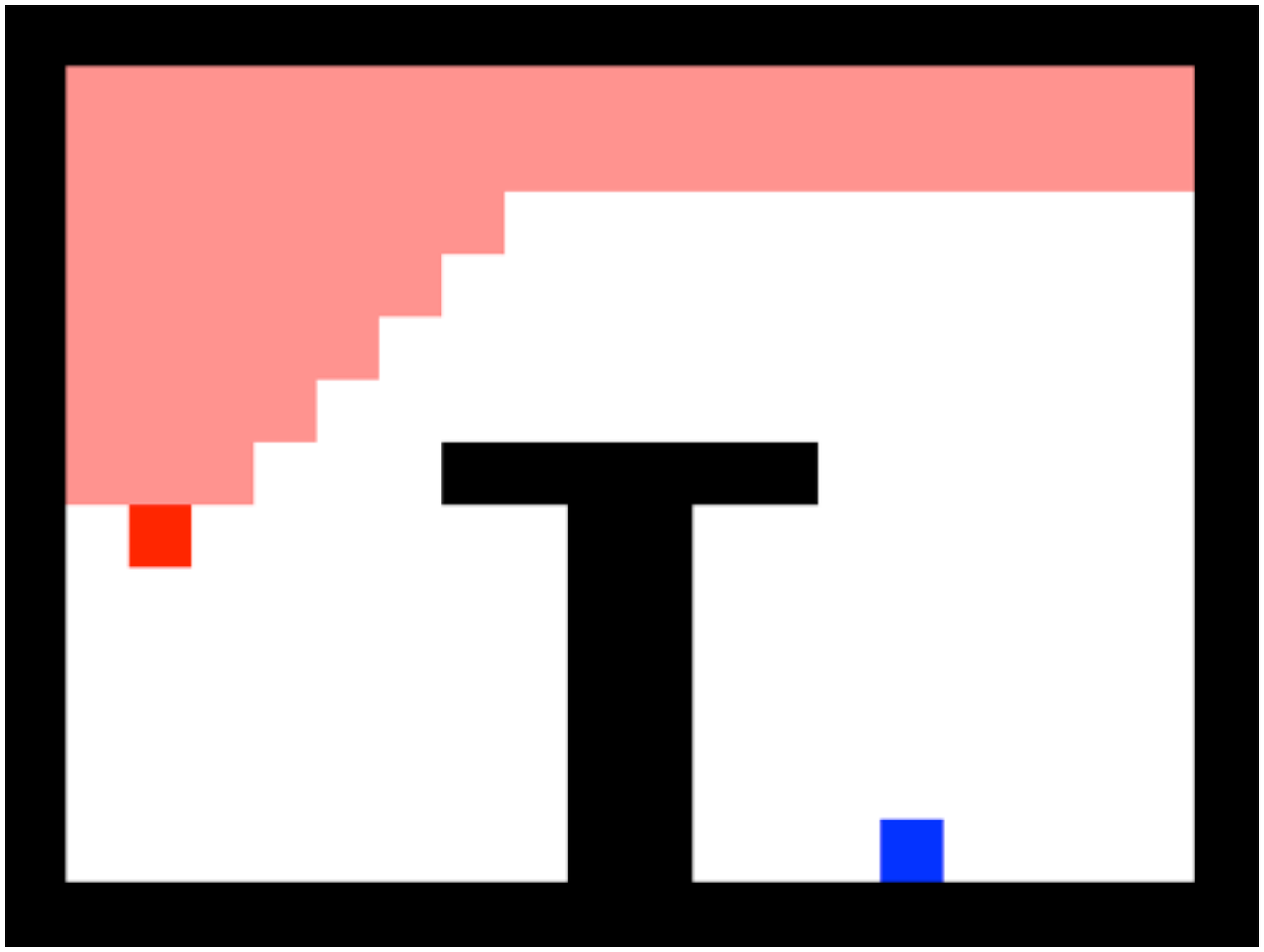
Take an ‘unsolvable’ level, and simulate playing it.



Mechanic Miner

Evaluation

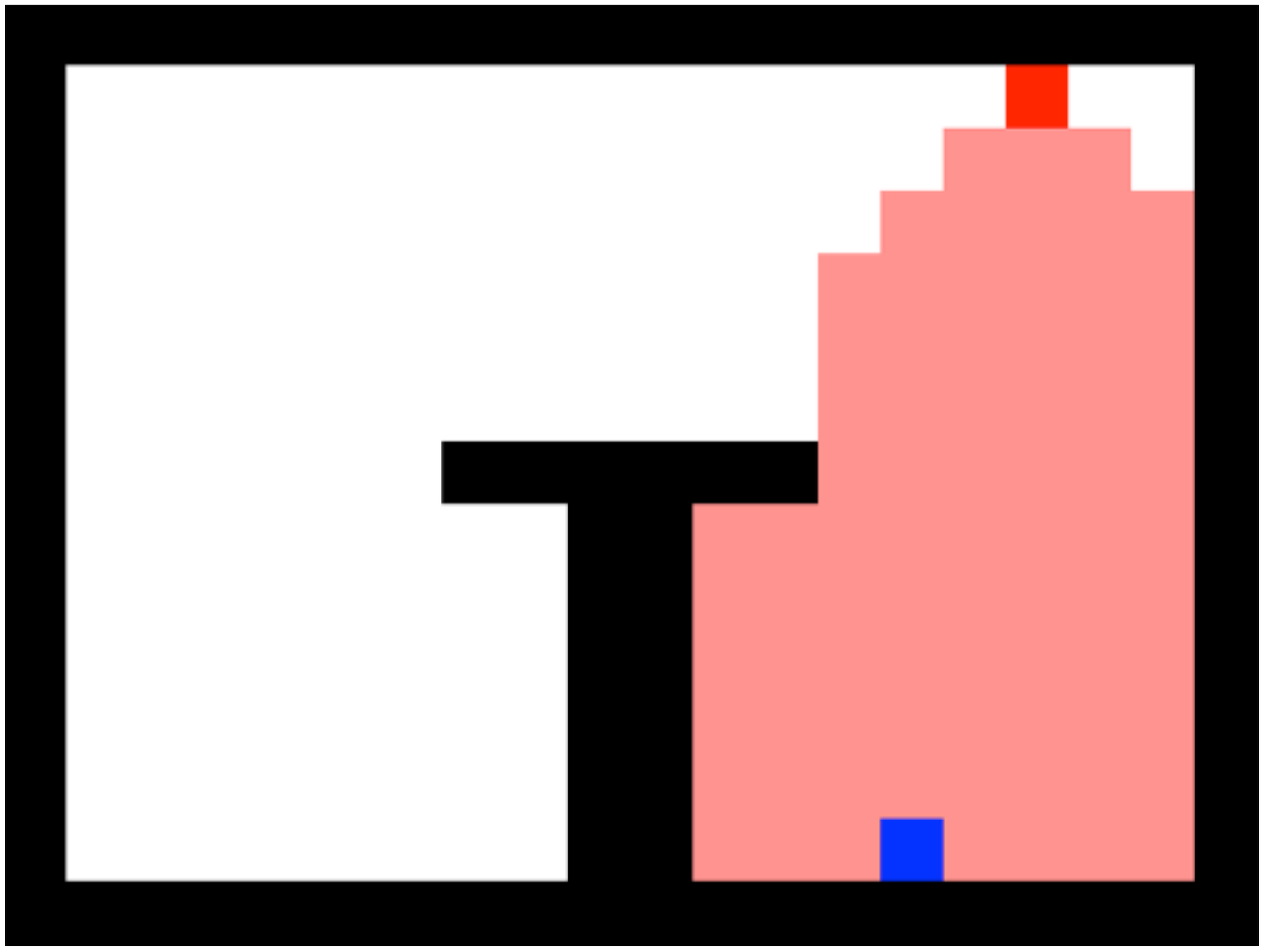
Take an ‘unsolvable’ level, and simulate playing it.



Mechanic Miner

Evaluation

Take an ‘unsolvable’ level, and simulate playing it.



Mechanic Miner

Example Mechanics (hot off the press!)

(as well as 1000 game-breaking, exception-throwing,
programmer-hating mistakes)

Mechanic Miner

Example Mechanics (hot off the press!)

```
INVERTSIGN Player.acceleration.y
```

```
(as well as 1000 game-breaking, exception-throwing,
programmer-hating mistakes)
```

Mechanic Miner

Example Mechanics (hot off the press!)

```
INVERTSIGN Player.acceleration.y
```

```
HALVE Player.y
```

```
(as well as 1000 game-breaking, exception-throwing,
programmer-hating mistakes)
```

Mechanic Miner

Example Mechanics (hot off the press!)

```
INVERTSIGN Player.acceleration.y
```

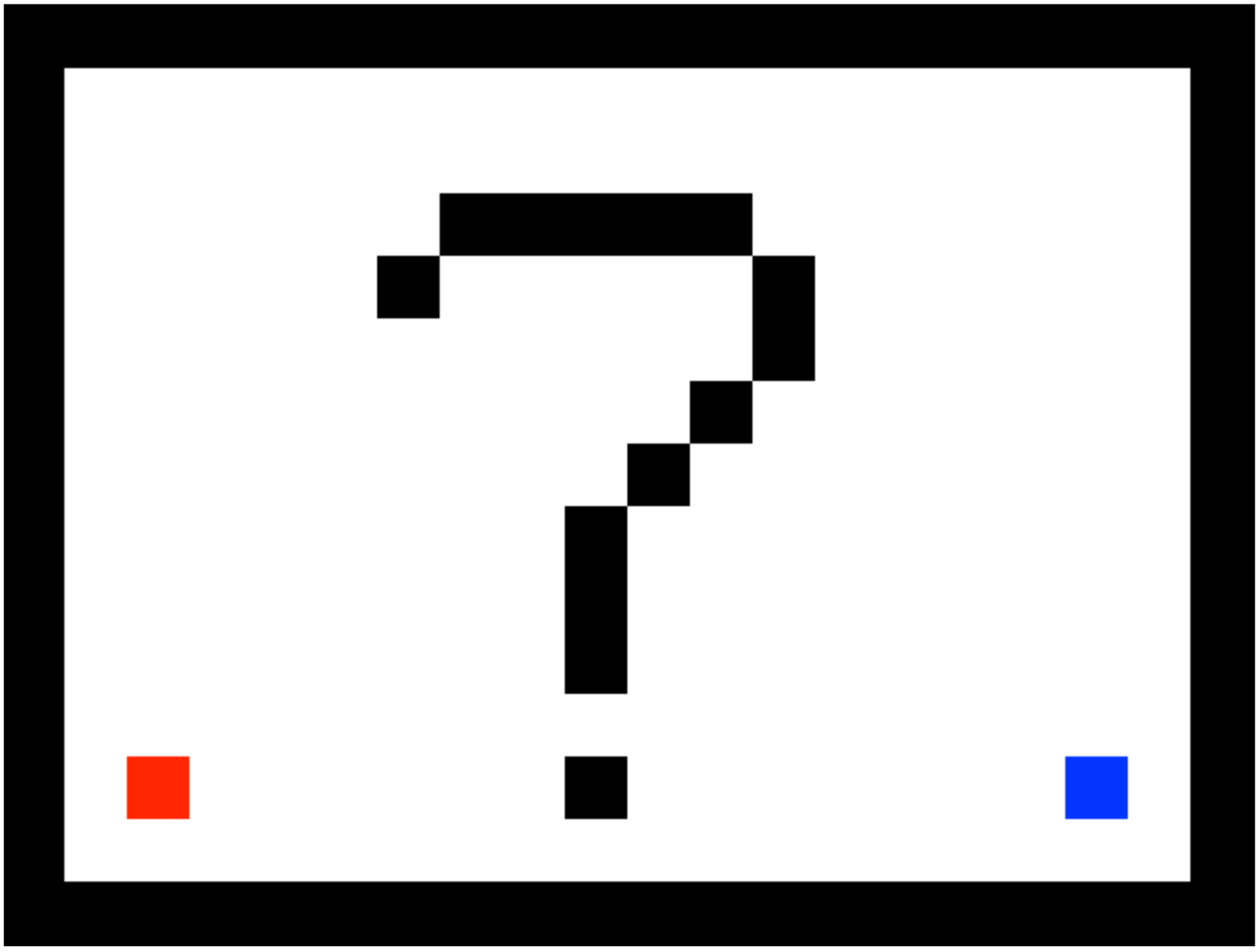
```
HALVE Player.y

HALVE FlxGame.timescale
```

(as well as 1000 game-breaking, exception-throwing, programmer-hating mistakes)

Mechanic Miner

Level Design



Puzzle Platformers

Sample Levels

Puzzle Platformers

Sample Levels

```
11111111111111111111
1                  1
1           X  11  1
1      1111111111  1
1                111
1                  1
1                  1
11111              1
11111              1
11111              1
11111      11      1
11111      11      1
1                  1
1        S         1
11111111111111111111
```

Puzzle Platformers

Sample Levels

```
111111111111111111111
1                     1
1                     1
1           X   11    1
1      1111111111     1
1                  111
1                     1
1                     1
1                     1
1                     1
11111                 1
1111 1                1
11111                 1
1111 1        11      1
11111         11      1
1                     1
1         S           1
111111111111111111111
```

```
11VVVVVVVV1VVVVVVV1111
1                     1
1               V     1 1
1         1     V       1
1   S     1     V       1
1         11111V        1
1         11111V        1
1         11111VVVVV    1
VVVVVVVVVVVV1           1
1      VVVVVVVVVVVVV    1
11        11111         1
11           X          1
11                      1
1                       1
1                       1
111111111111111111111111
```

Mechanic Miner

Exciting things!

- Levels that are solvable using mechanic A, but *not* using mechanic B.
- Levels that are solvable only by using mechanics A *and* B.
- Levels that require *x* steps of planning from the player.

Puzzle Platformers

Future areas...

- Richer code synthesis (component creation, mechanical constraints)
- Better fitness functions for mechanics and levels
- Integration into non-puzzle contexts, e.g. ANGELINA's Metroidvania games
- More game engines, more genres, more more.

Questions!

More on ANGELINA and play games:
http://www.gamesbyangelina.org

@mtrc **@angelinasgames**

More on Computational Creativity at Imperial:
http://ccg.doc.ic.ac.uk