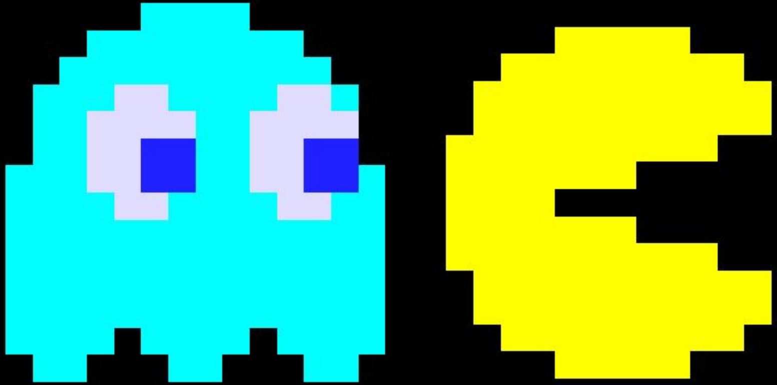


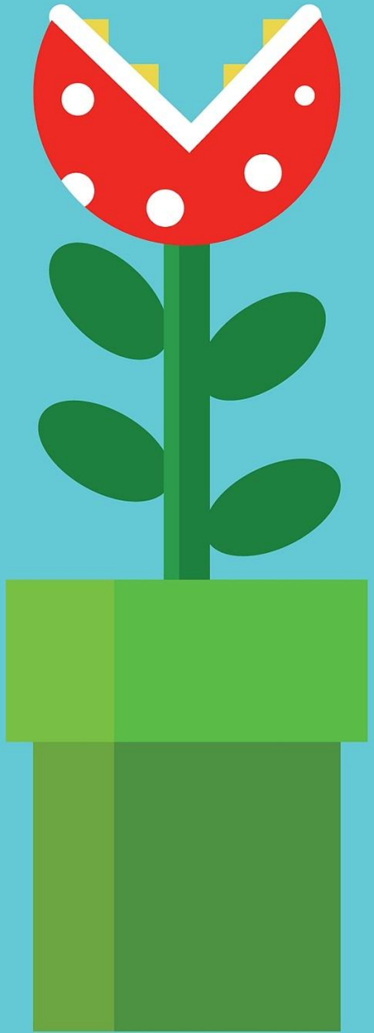
Cindy Wang
Steven Shao
Yuncheng Jiang

MakerGame

Game Programming Language

Outline

- Motivation
- Features
- Runtime
- Architecture & Tests
- Demo



Motivation

Game Maker

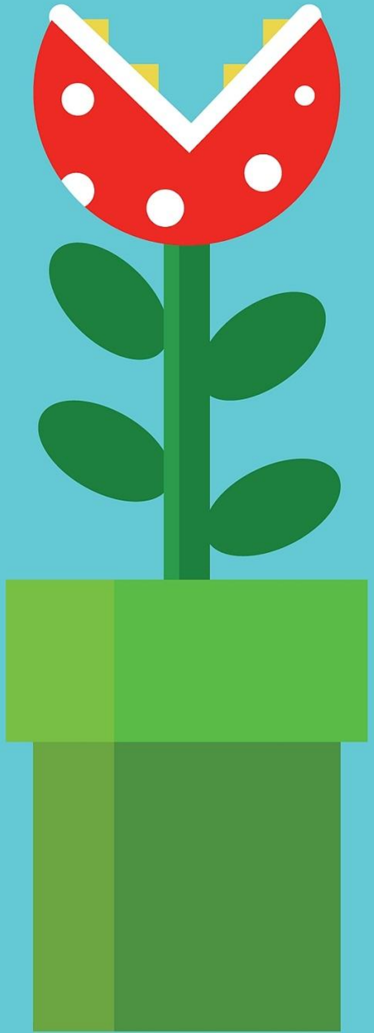
- Game asset management
- Graphics, sounds, input built in
- Entity resource handling
- Execution flow following object lifetimes

C/C++

- A real programming language
- Arbitrary data structures - arrays
- Object & Library encapsulation - methods, namespaces
- Fast - a blank slate

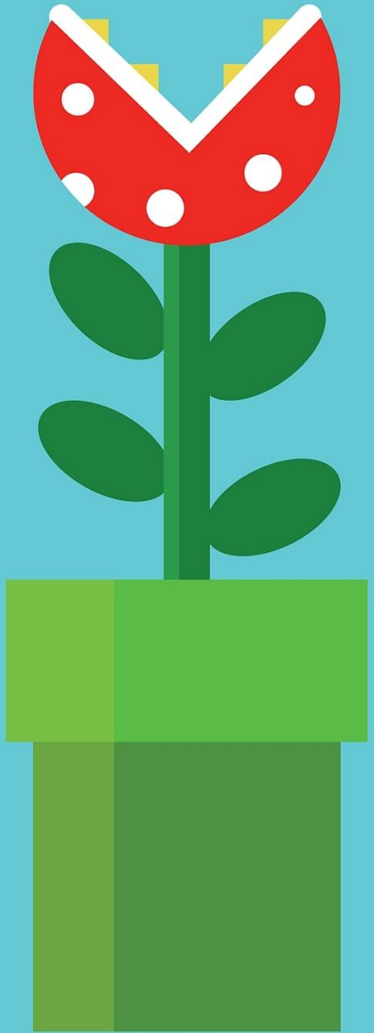
Motivation

- **GameMaker for programmers**
- **Gems from both worlds**
    - General collections
    - Objects & inheritance
    - Event-driven
    - Standard library & types for games



Features

- C with "objects"
- Entity life & event handling
- External linking & libraries



C with Objects - Types & Functions

```
int x = 3 + 5;
float pi = 3.14;
bool answer = true;

int table[2][3] = [[1,2,3],[4,5,6]];
string file = "player.png";

sprite p = std::spr::load(file);
sound q = std::snd::load("bonk.ogg");
```

```
int square(int x) { return x * x; }
int sum(int x[2]) { return x[0] + x[1]; }
```

C with Objects - Control Flow

```
if (x < 100) x += 3;
else { x = 100; hit_end = true; }

while (!settled) { moveDown(); }

{ int x = 3; }

for (int i = 0; i < n; ++i) {
  sum += i;
  if (tooHigh(sum)) break;
}
```

C with Objects - Objects

Definition

```
object Player {
  int x; int y; ...
  int getHealth()   { ... }
  ...
  event create(...) { ... }
  event step        { ... }
  event draw        { ... }
  event destroy     { ... }
}
```

Manipulation

```
void doStuff(Enemy e) {
  object o = none;

  Player p = create Player(...);
  object m = p;
  int y = p.getHealth();

  p.x = 3;

  if (p == o) { ... }

  destroy e;
}
```

C with Objects - Inheritance

Definition

```
object Enemy {
  int health; sprite s;
  bool touchingPlayer() { ... }
  event create(int hp)  { ... }
  event step            { ... }
  event draw            { ... }
  event destroy         { ... }
}
```

Inheritance

```
object Missile : Enemy {
  event create() {
    super(100);
    s = spr::load("missile.png");
  }

  void explode() { ... }

  event destroy() {
    if (touchingPlayer())
      explode();
    super();
  }
}
```

C with Objects - Modules

Nested Namespaces

```
namespace math {
  int square(int x) { ... }
  extern float sin(float x);
}
```

Access Levels

```
namespace spr {
  private namespace p {
    extern sprite load_sprite(...);
  }
  sprite load(...) { ... }
}
... { spr::p::load_sprite(...); }
```

Files & Scope

```
// from MAKERGAME_PATH
namespace math = open "math.mg";
namespace spr  = open "spr.mg";
using math;

... {
  ...
  sprite s = spr::load(...);
  int x = pi;
  float y = sin(5);
  ...
}
```

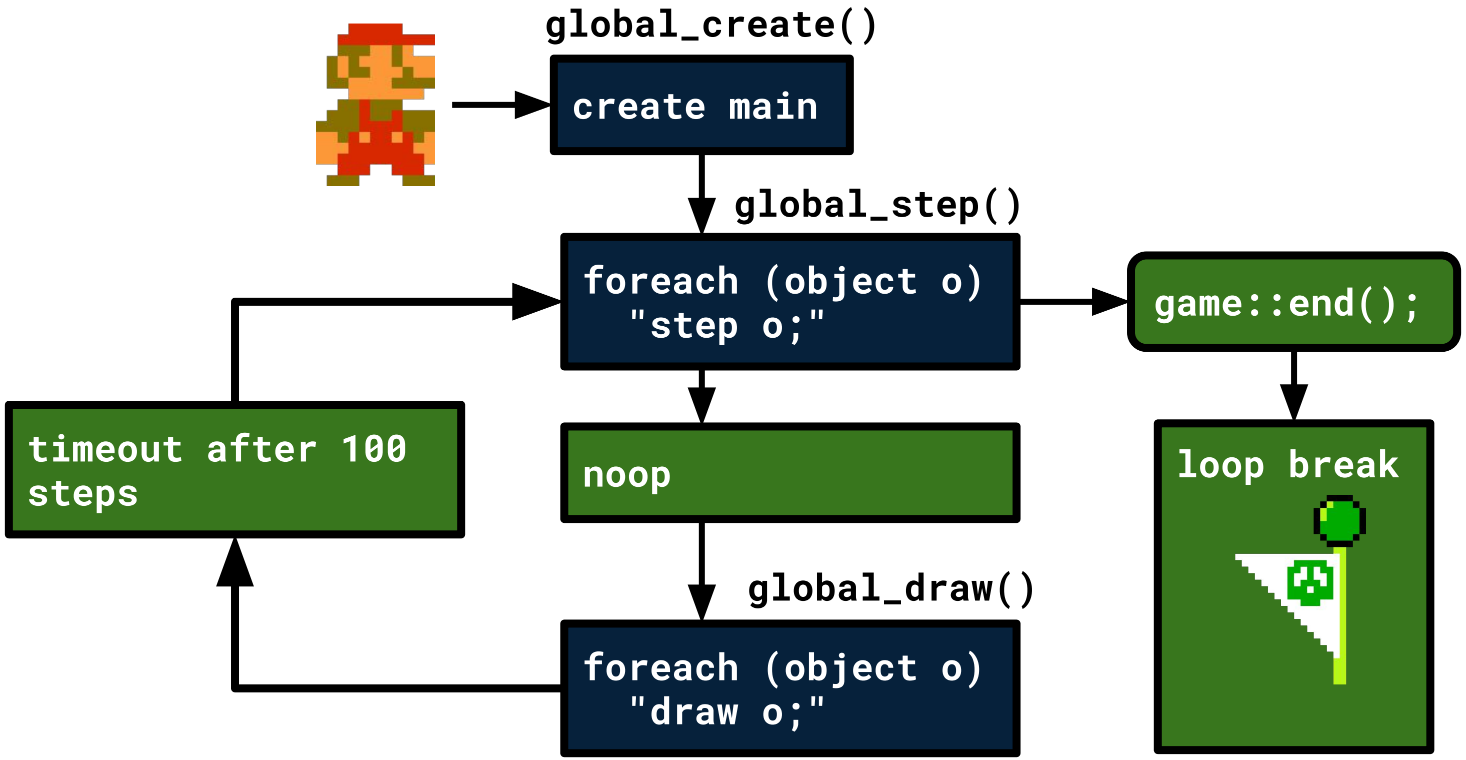
Entity Life & Event Handling

Key Operation: Iteration

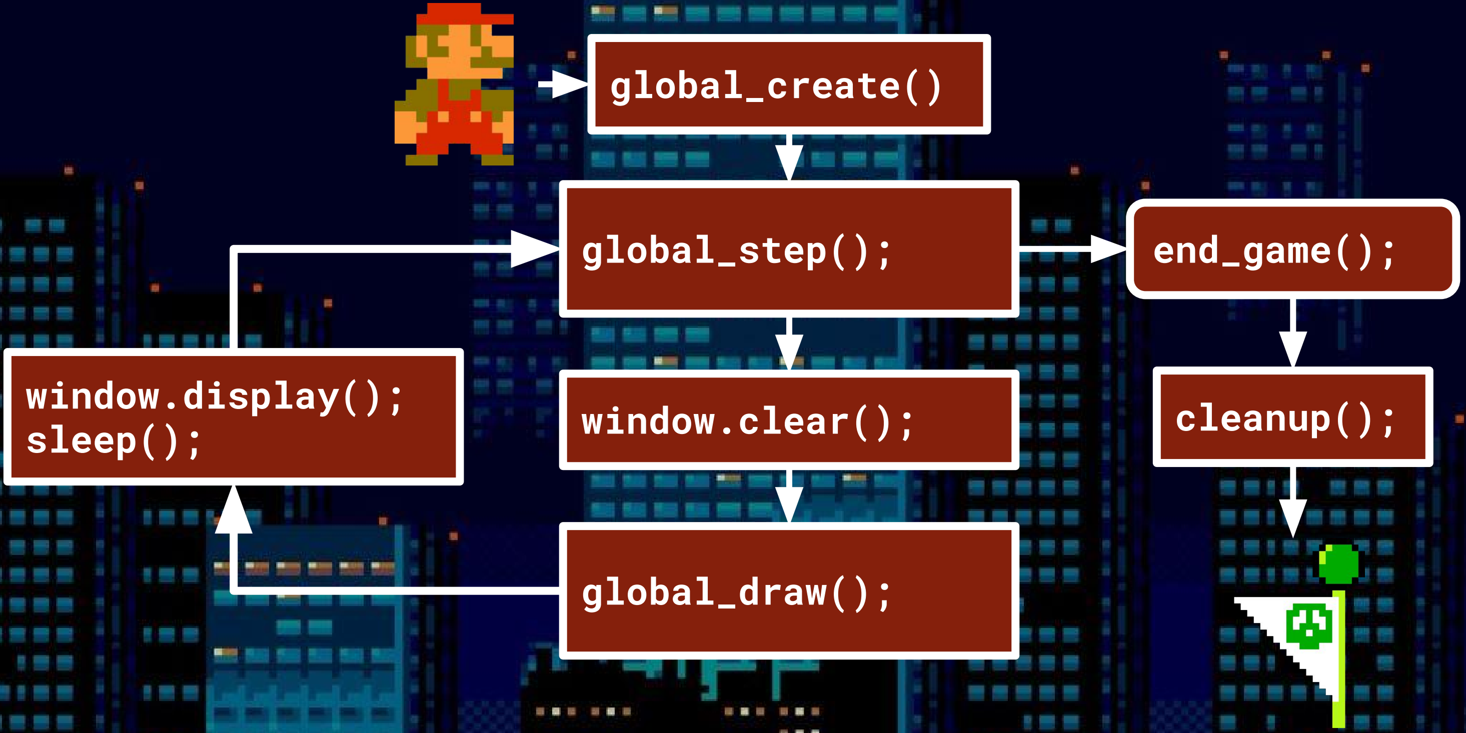
```
foreach (Enemy e) {
  if (colliding(e, this)) {
    --health;
    destroy e;
  }
}
```

Examples

```
void destroyAllButMe(object m) {
  foreach (object o) {
    if (o != m) destroy o;
  }
}

bool isAlive(object m) {
  foreach (object o)
    if (o == m) return true;
  return false;
}
```


global_create()
create main
global_step()
foreach (object o) "step o;"
game::end();
timeout after 100 steps
noop
loop break
global_draw()
foreach (object o) "draw o;"

Life & Event Handling: Runtime



```
global_create()
```

```
global_step();
```

```
window.clear();
```

```
global_draw();
```

```
window.display();
sleep();
```

```
end_game();
```

```
cleanup();
```

Life & Event Handling: Runtime

Sample functions:

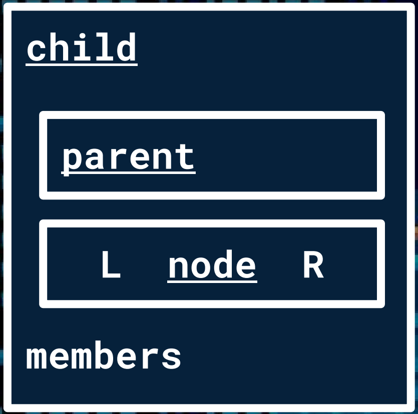
```
void printb(bool b)
void print(int x)
...

sf::Sound *load_sound(...)
sf::Sprite *load_image(...)
void draw_sprite(sf::Sprite *, ...)
void play_sound(sf::Sound *, ...)
...

bool key_pressed(int code) { ... }
...

void set_window_size(...)
void set_window_clear(...)
void end_game()
```

Accessing from MakerGame

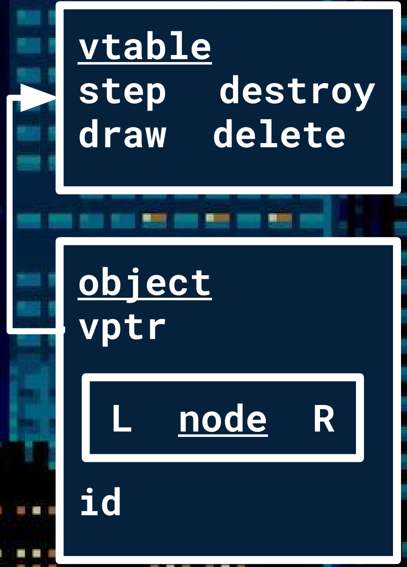
```
extern sound load_sound(...);

{
  sound s = load_sound(...);
}

(extern definitions in std.mg)
```

Life & Event Handling: Objects

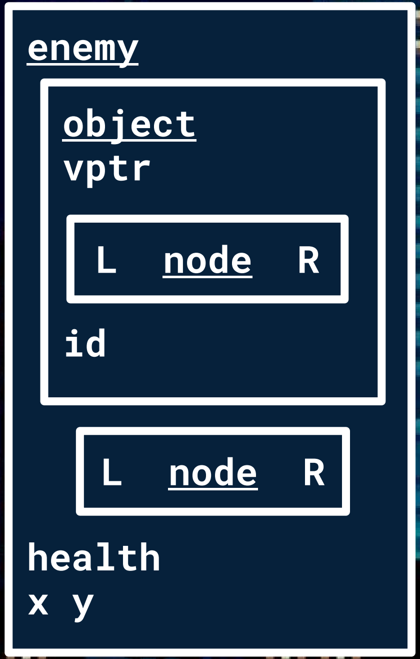
Anatomy of an object



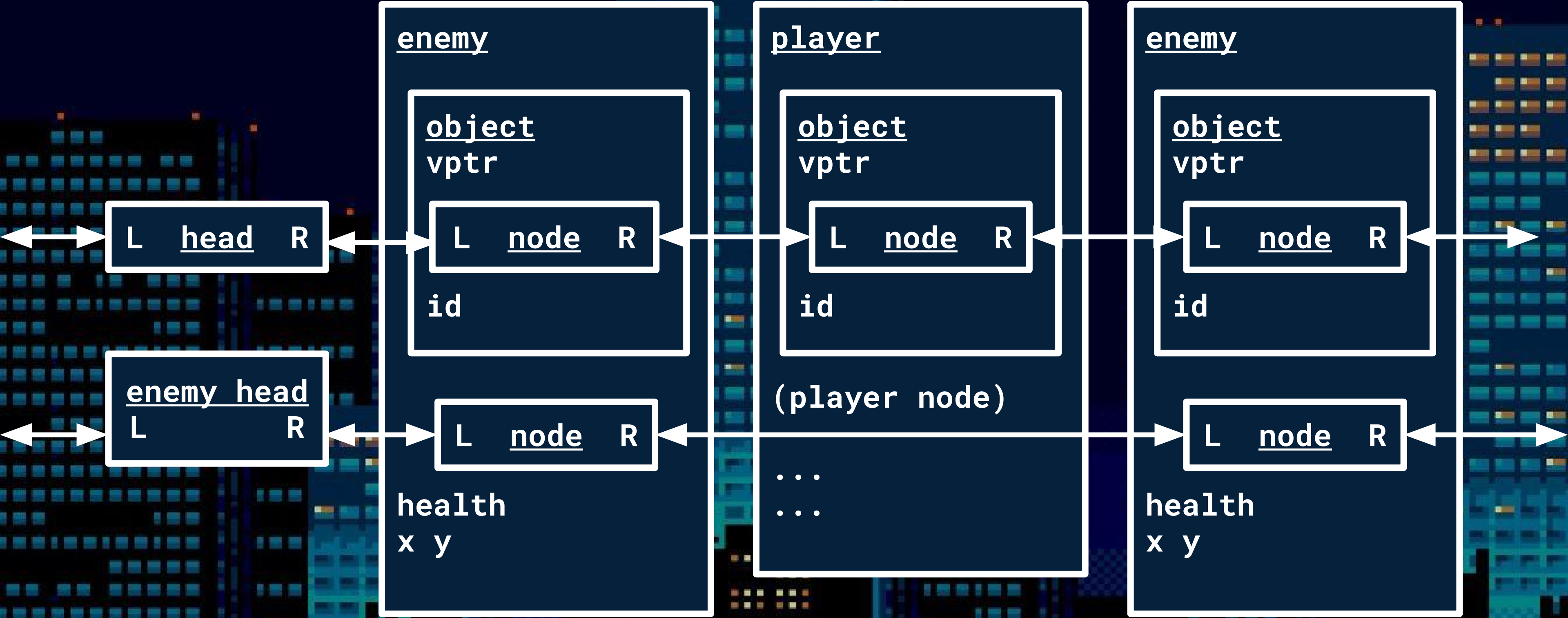
The universal parent



Example

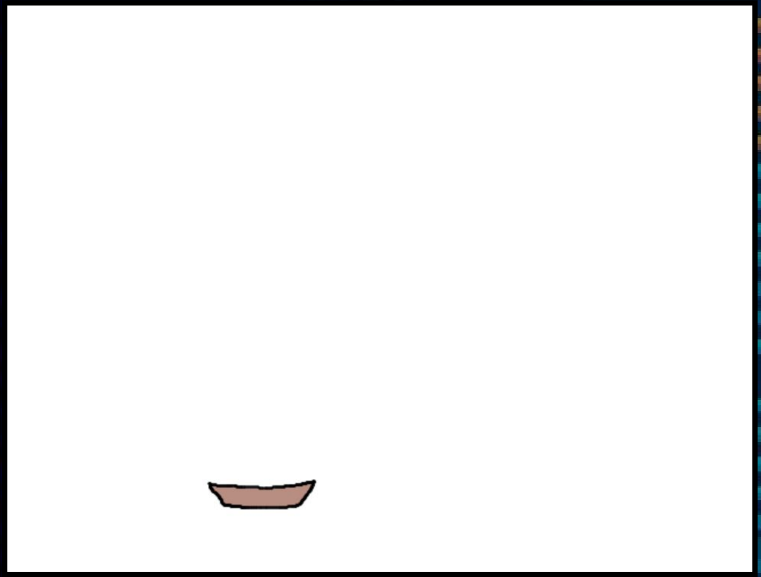


Life & Event Handling: Objects

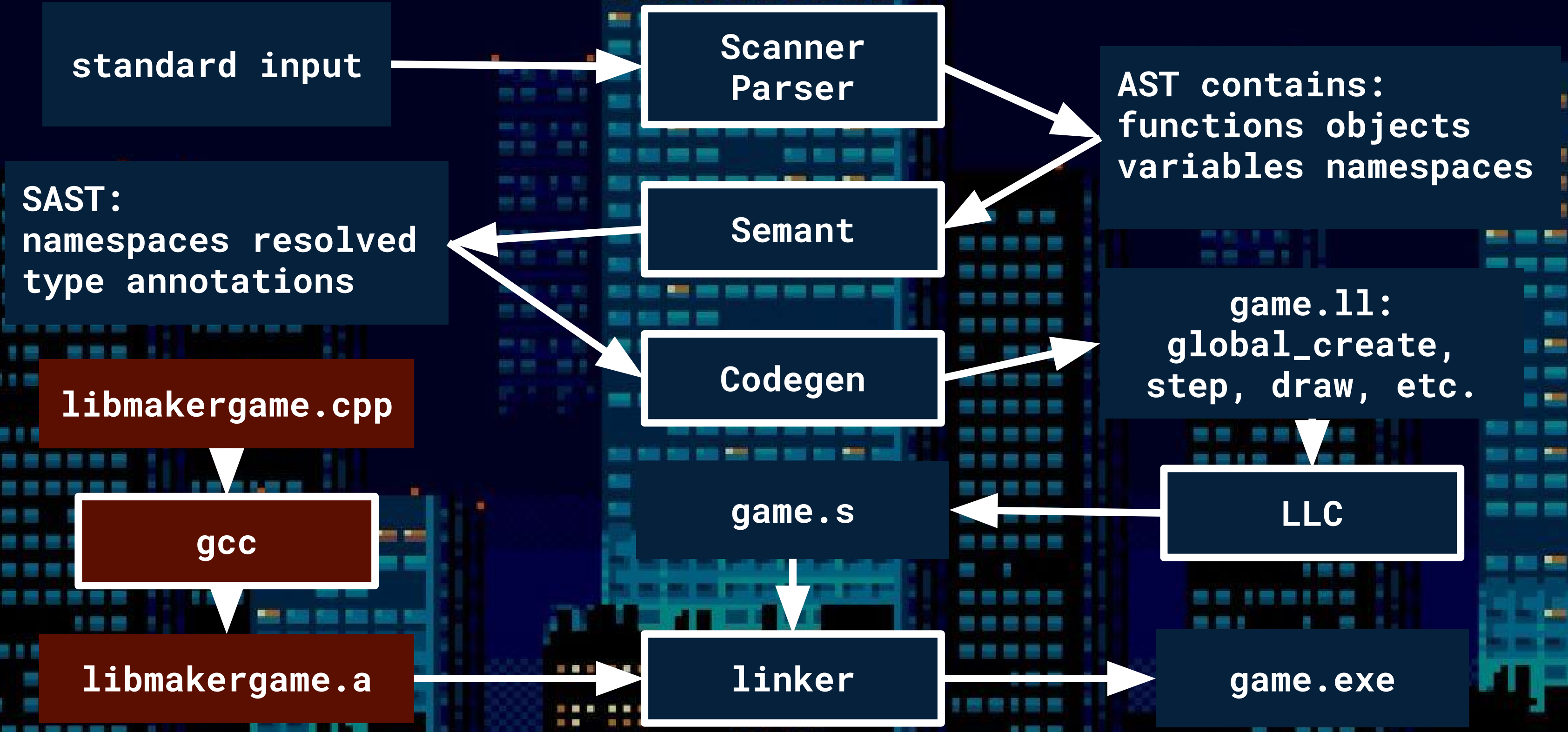


Full Sample Program

```
object Player {
  sprite spr; int x; int y;

  event create {
    spr = spr::load("res/player.png");
    x = 350; y = 500;
  }

  event step {
    if (key::is_down(key::Left)) x -= 5;
    if (key::is_down(key::Right)) x += 5;
  }

  event draw { spr::render(spr, x, y); }
}

object main { event create { create Player; } }
```

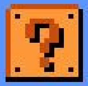


Compiler Architecture

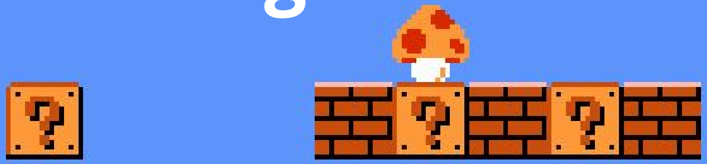

standard input
Scanner Parser
AST contains: functions objects variables namespaces
Semant
SAST: namespaces resolved type annotations
Codegen
game.ll: global_create, step, draw, etc.
LLC
game.s
libmakergame.cpp
gcc
libmakergame.a
linker
game.exe

Testing: The tests

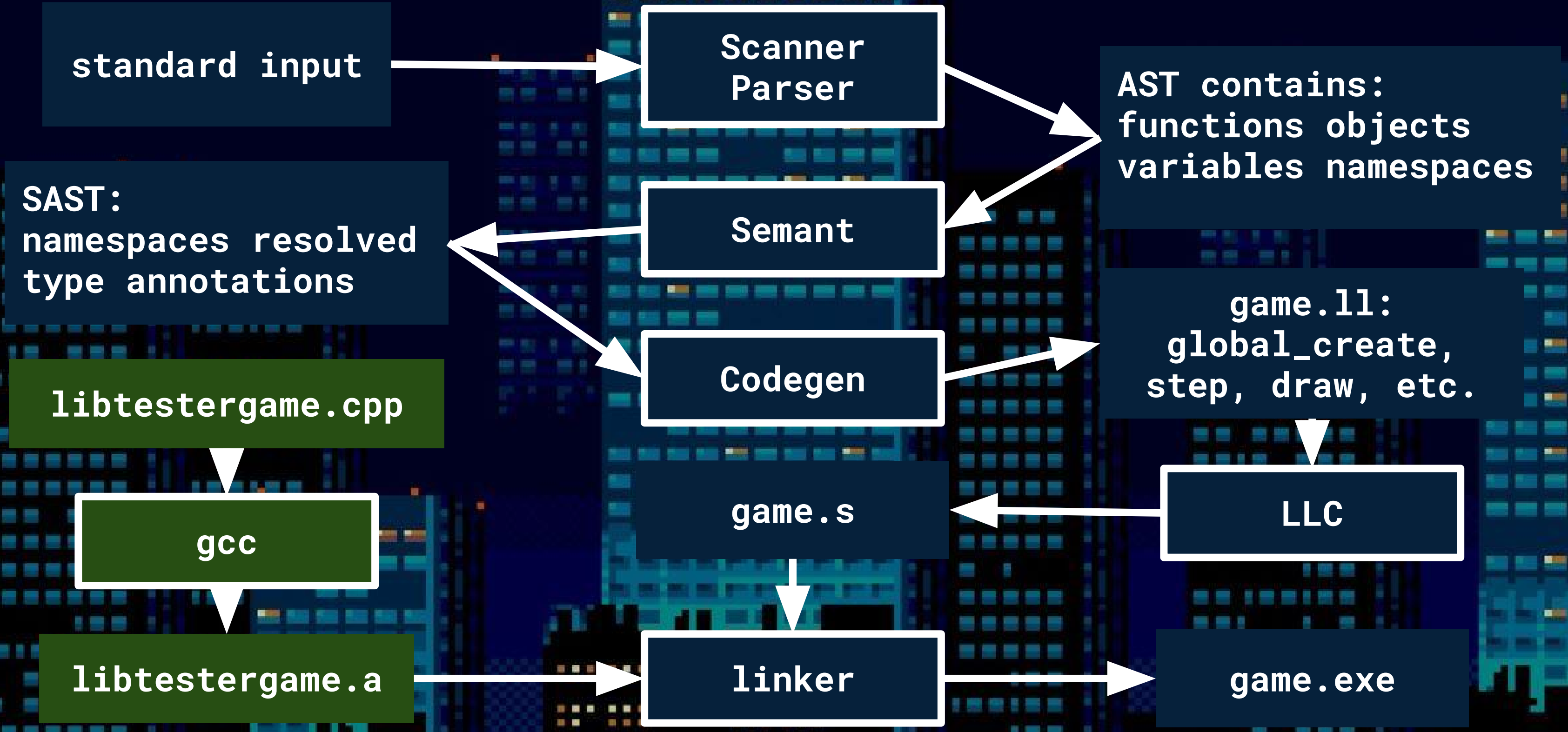
- **Unit tests - language features**
  - 410 tests



- **Games - runtime, stress testing**
  - Egg drop
  - Tetris

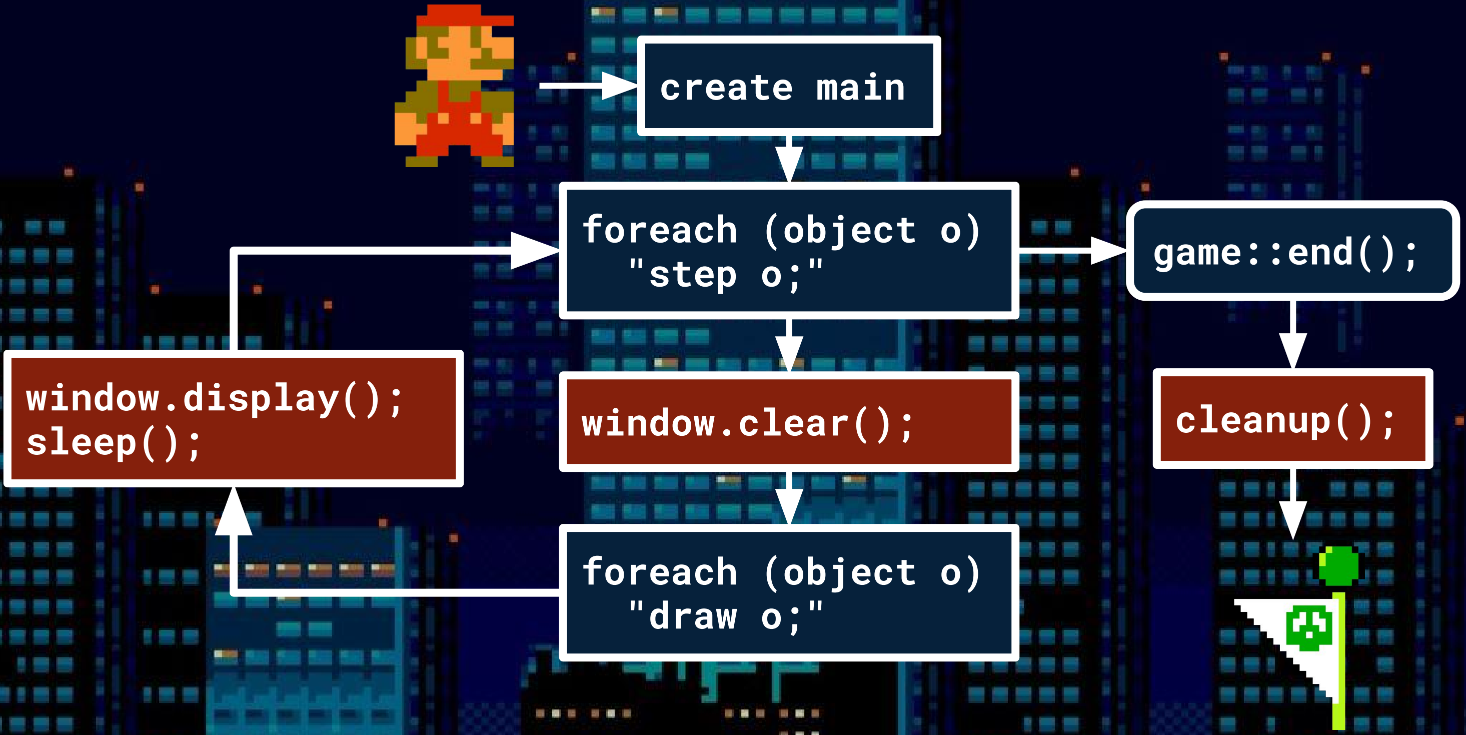


Testing: Compiler Architecture

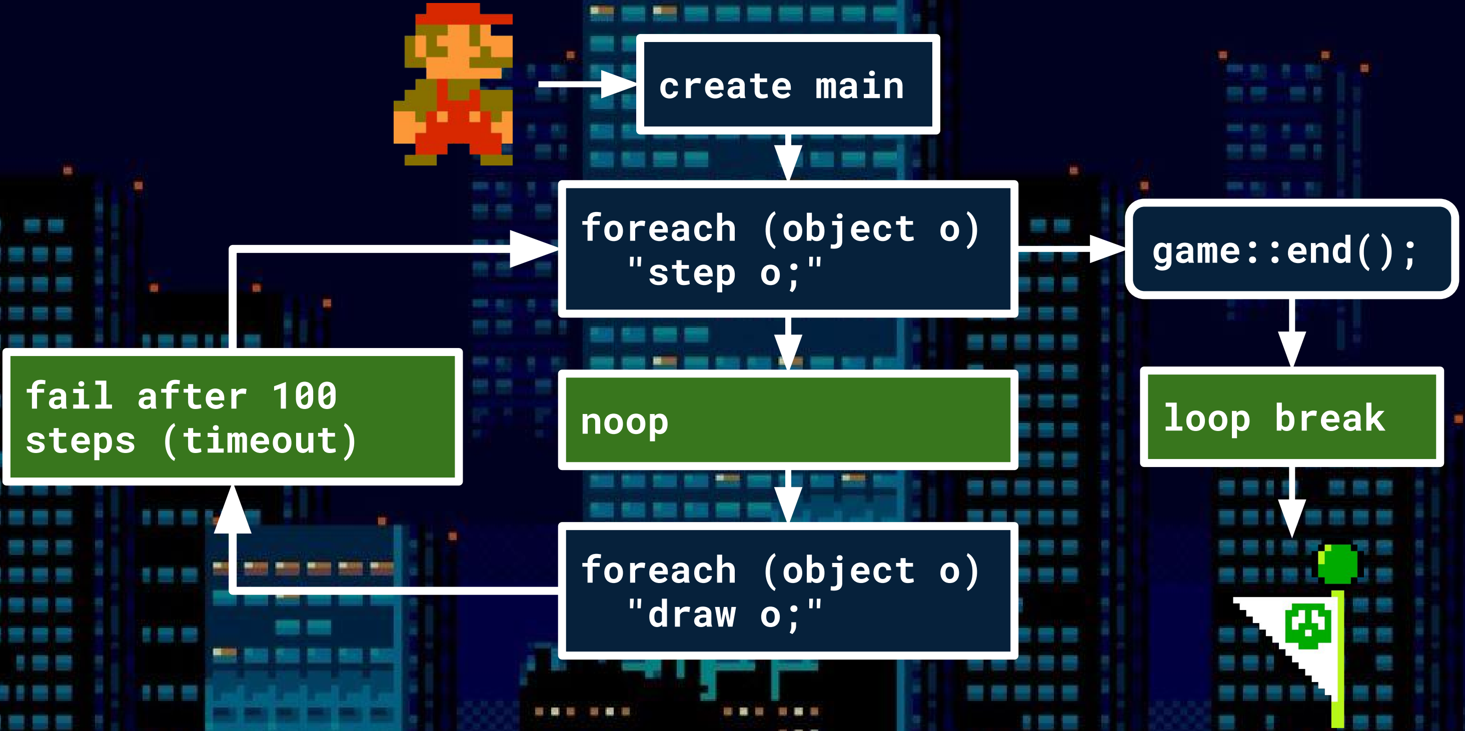


standard input
Scanner Parser
AST contains: functions objects variables namespaces
Semant
SAST: namespaces resolved type annotations
Codegen
game.ll: global_create, step, draw, etc.
LLC
game.s
libtestergame.cpp
gcc
libtestergame.a
linker
game.exe

Testing: Unit Test Framework



Testing: Unit Test Framework

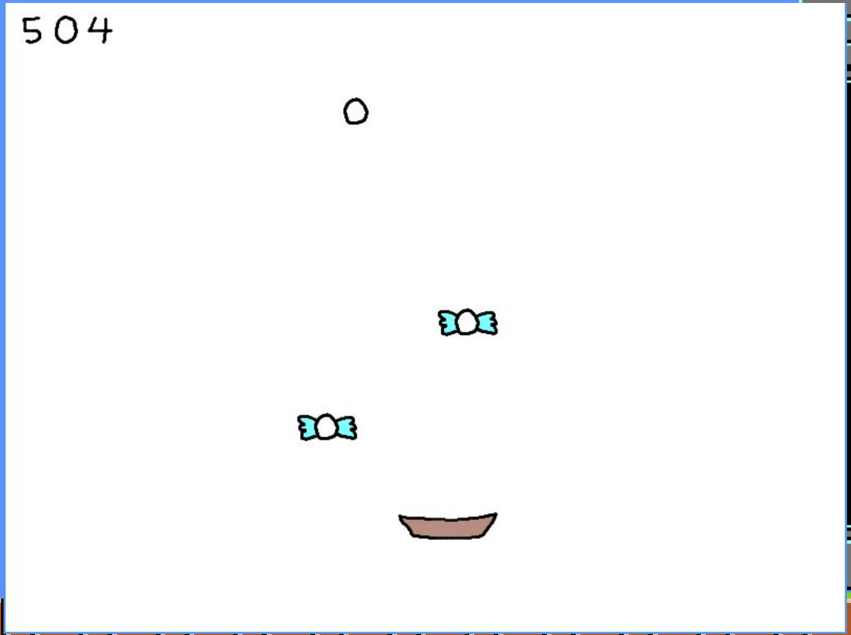

create main
foreach (object o) "step o;"
game::end();
noop
loop break
fail after 100 steps (timeout)
foreach (object o) "draw o;"

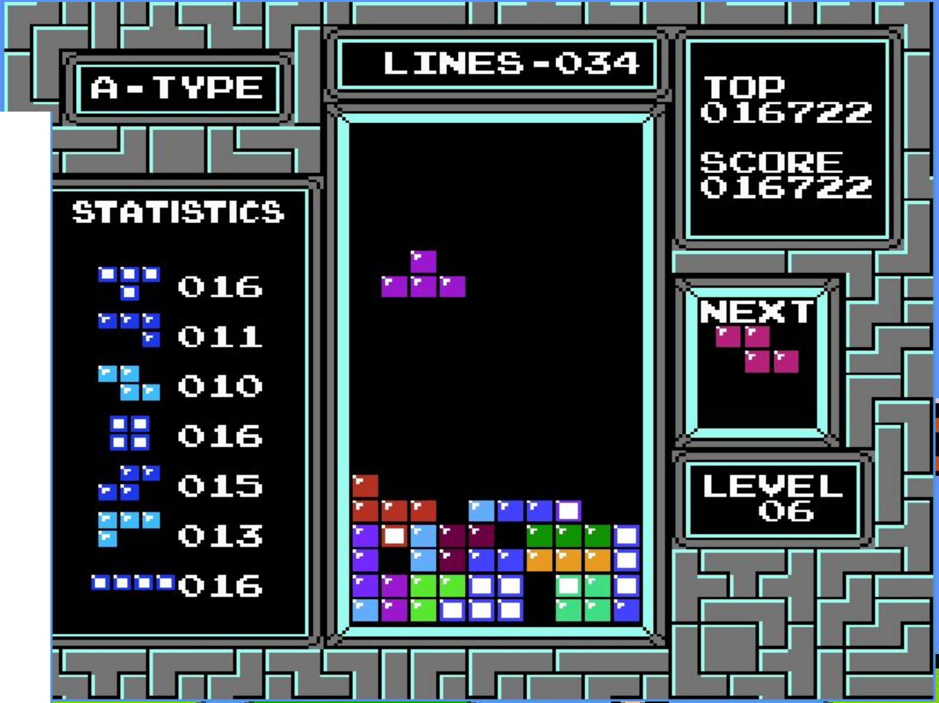
Testing: Unit Tests

```
// basic features: arrays
int make_ten_of[10](int x) {
  int ret[10];
  int i;
  for (i = 0; i < 10; ++i)
     ret[i] = x;
  return ret;
}

object main {
  event create {
     int i = 3;
     int j[10] = make_ten_of(5);
     std::print::i(j[i]);
     std::game::end();
  }
}
```

```
// complex game loop cases
object parent { void detonate() { destroy this; } }
object child : parent { }
object main {
  int j;
  event create { j = 0; }
  event step {
     for (int i = 0; i < 10; ++i) create child;
     for (int i = 0; i < 10; ++i) create parent;
     int i = 0;
     foreach (child c) c.detonate();
     foreach (parent c) ++i;
     std::print::i(i);
     ++j;
     if (j >= 6) std::game::end();
  }
}
```

Testing: Demos





Thank you!
Questions?