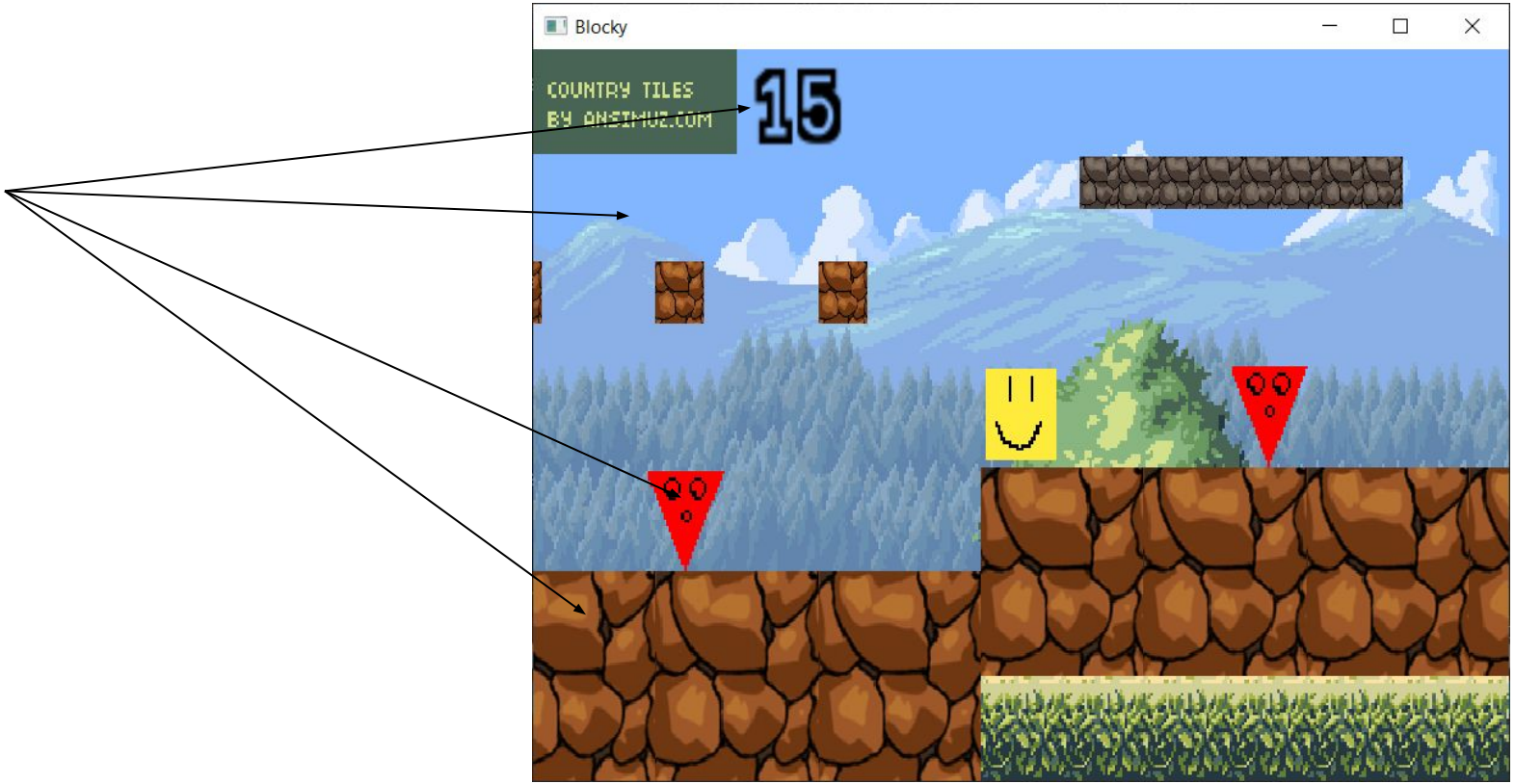
Blocky Stickman

A Holy War of Geometry?

Natalie Dorshimer and Lucas Campbell

A Game, and a Game Engine

Entities!

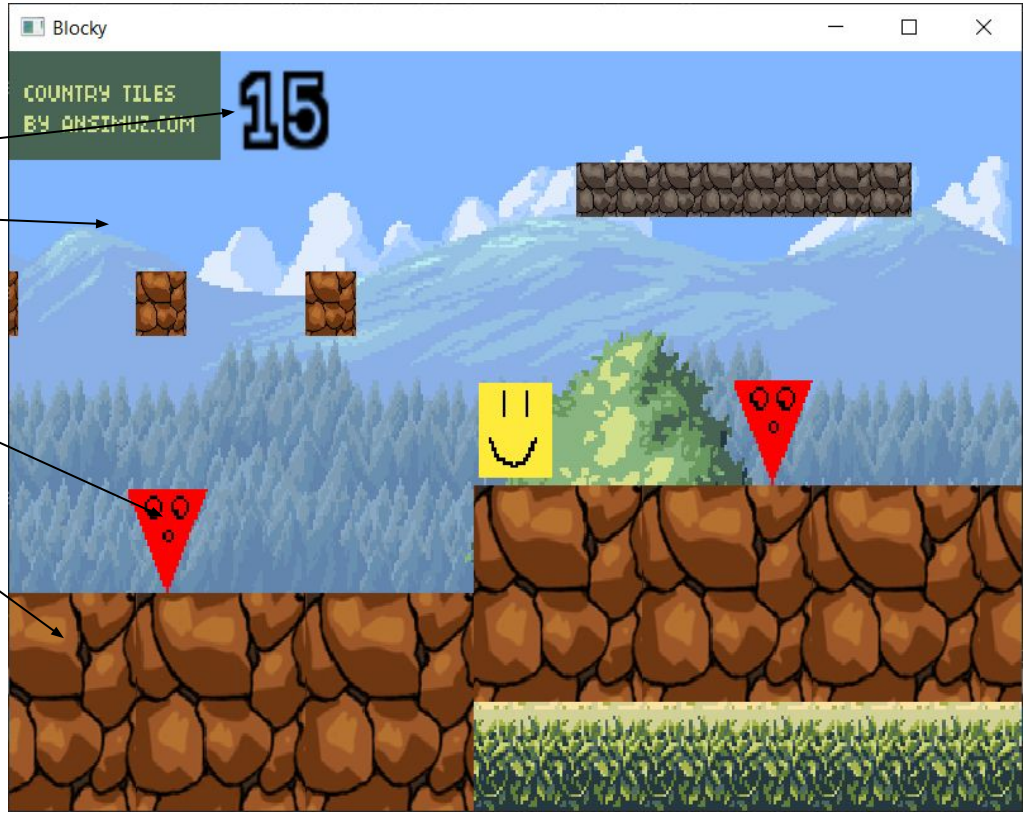


A Game, and a Game Engine

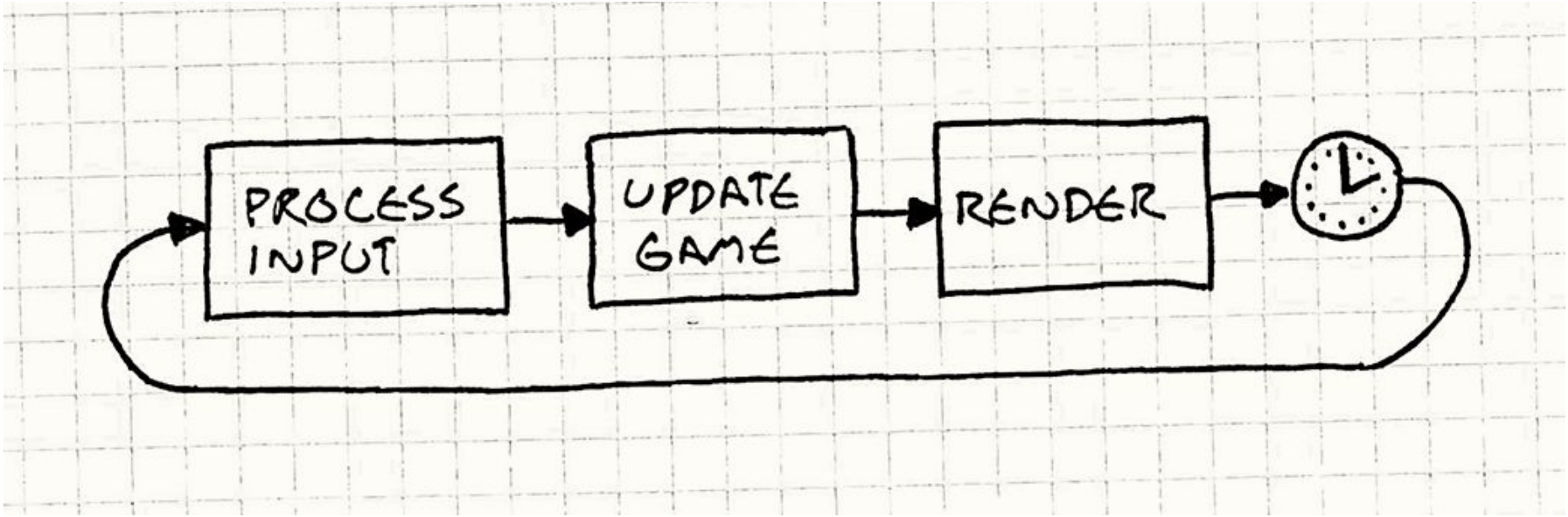
Entities!

with Components:

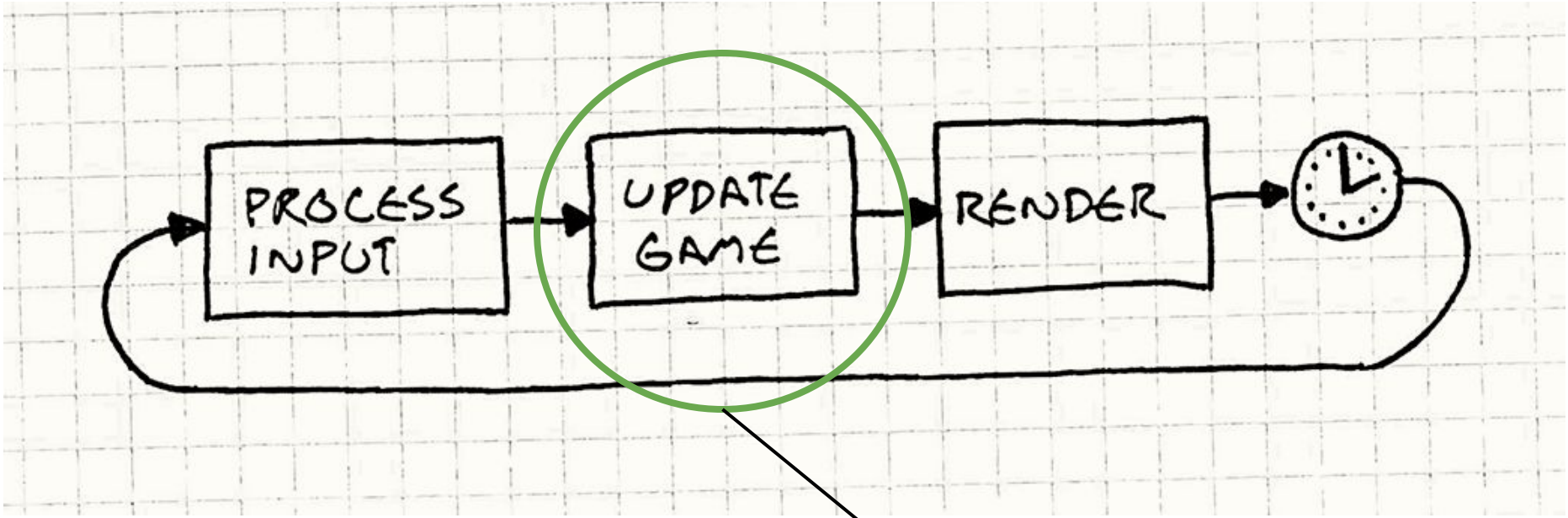
- SpriteComponent
- BackgroundComponent
- TextComponent
- TimerComponent
- etc



Let’s Back Up: Traditional Game Loop

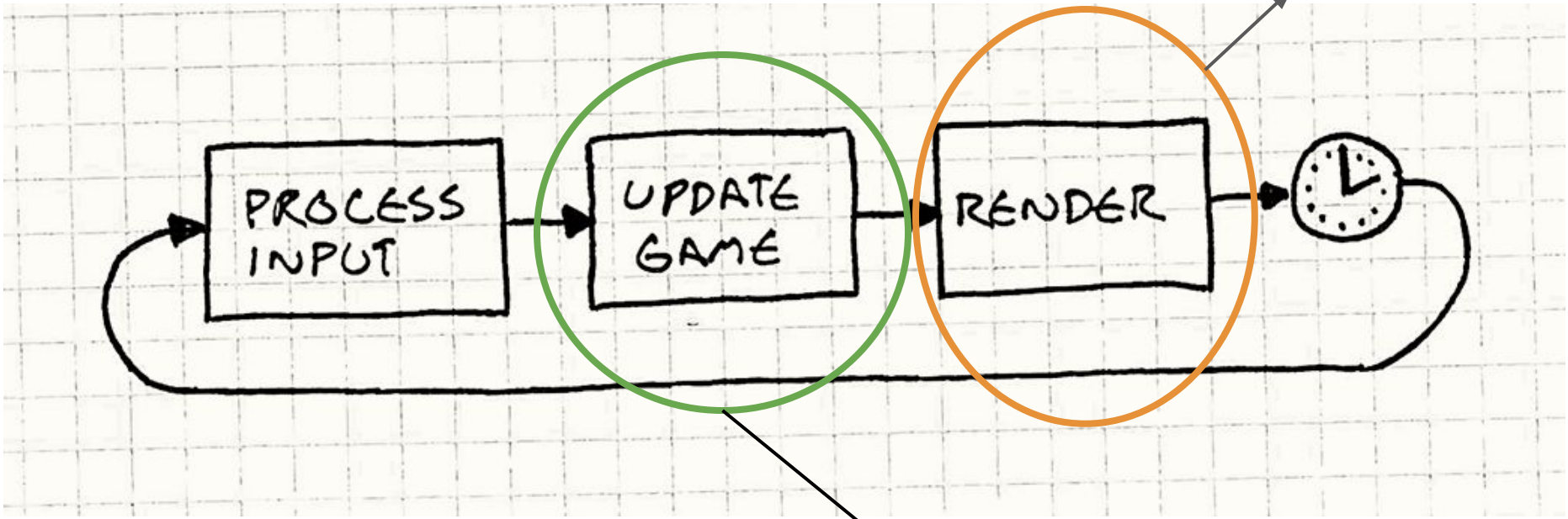


Let’s Back Up: Traditional Game Loop



Lots of processing (of course!)

Let’s Back Up: Traditional Game Loop



One way: every sprite for itself

```
//pseudocode

World w;

while (true) {
        input = getUserInput();
        for (object in world) {
                object.update(input, w);  // handles collisions, player input,
                                          // movement, anything!
                                          // Requires object inheritance: very hard to maintain
                                          and scale for a wide variety of things
        }
        window.render();    //draws everything to the screen
}
```

This is a lot to process and think about for each object!

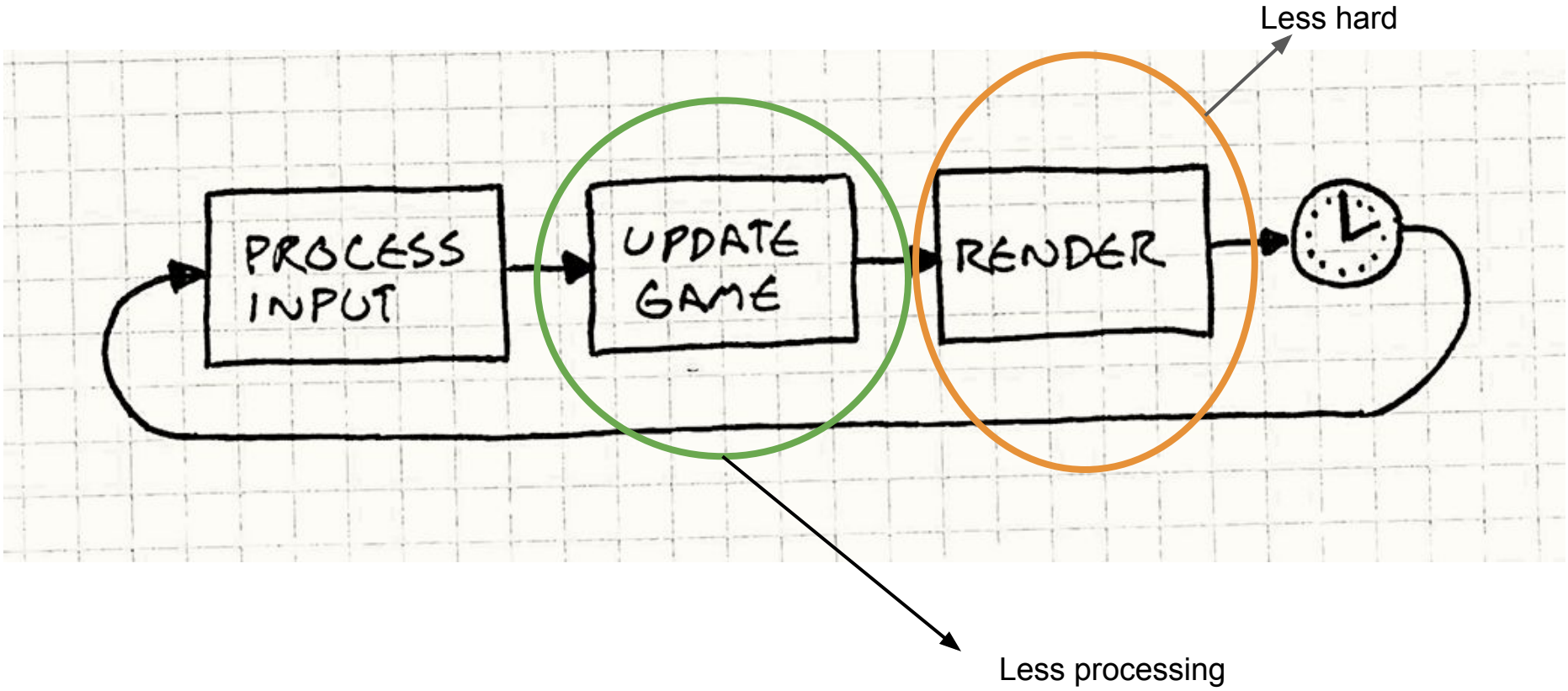
For example, collisions:

- Need to calculate if there was any collision with any other sprite, *for each sprite!*
  - O(n^2) :(

Introducing … Entity Component Systems (ECS)

- Now, we have a Registry to keep track of all of the things:
- Similar to a hashtable, but with much greater functionality!
  - Can get a list of entities with certain components (all entities with Sprites and Collider info, for example)
    - auto view = registry.view<Sprite, Collider>(); // list of colliding, sprite things!
  - Or, can access a specific component, given the entity
    - auto& sprite = view.get<Sprite>(entity); // get the sprite for that entity!

Now, we can sort in the Update and Render Steps!



New Solution with ECS:

- Outside *systems* do the work on subsets of entities,
  - Instead of each part of the world possibly worrying about every other part

- E.g.: Given some dt (time elapsed), we can make a simple system to model Translation, or movement in the world:

    for all entities with Position & Velocity components:

        Position.xPos += Velocity.dy * dt

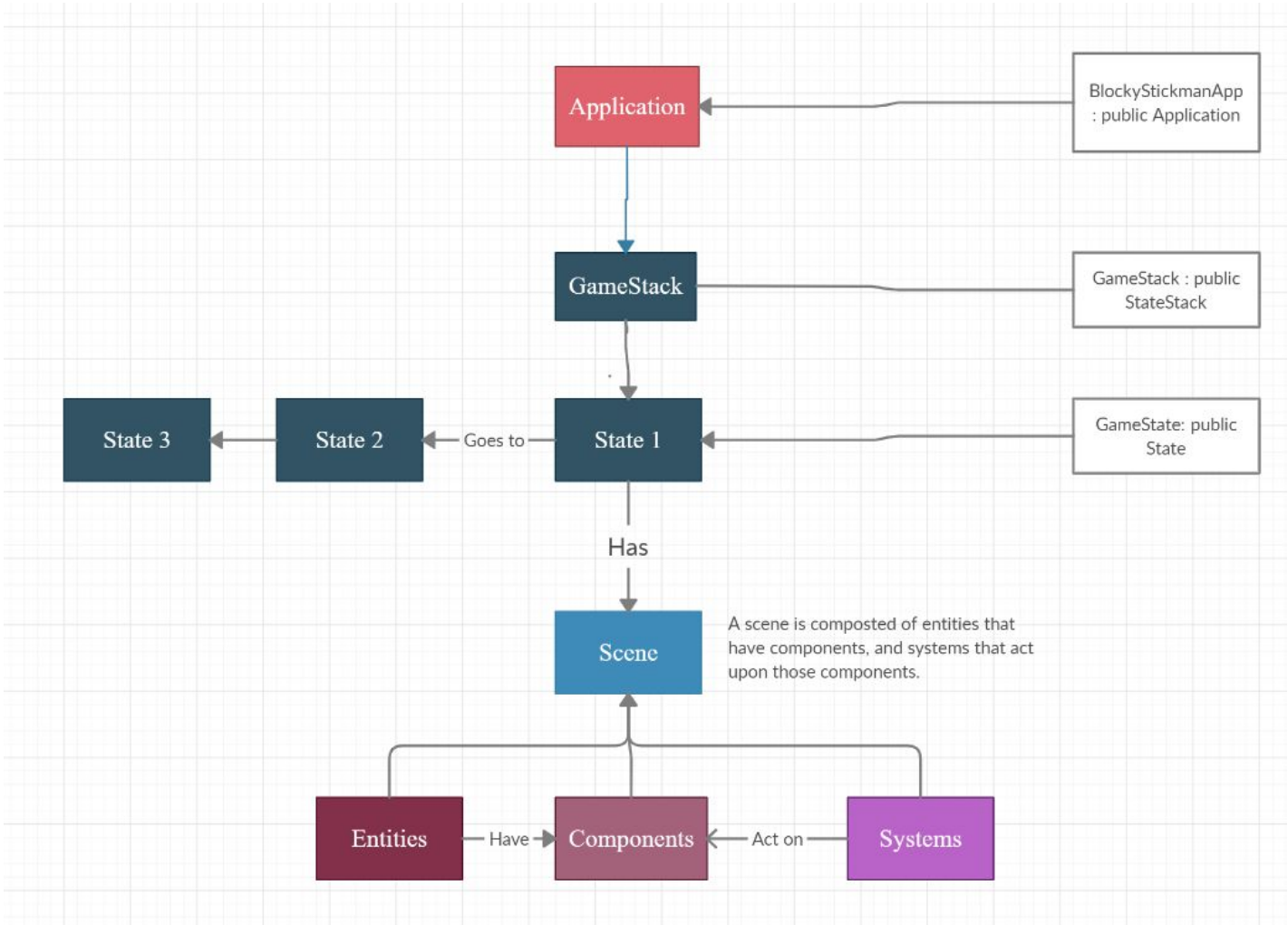
        Position.xPos += Velocity.dy * dt

```cpp
//Updates velocity and position of all entities that have velocity and sprite components
void GameSystems::TranslationSystem(Timestep dt, entt::registry& registry)
{
    auto view = registry.view<SpriteComponentPtr, VelocityComponentPtr>();

    for (auto ent : view)
    {
        auto& [sprite, velocity] = view.get<SpriteComponentPtr, VelocityComponentPtr>(ent);

        auto dx = velocity->velocity.x * dt; // dx/dt * dt = dx
        auto dy = velocity->velocity.y * dt;

        sprite->sprite.move(dx, dy);
    }
}
```


Application
BlockyStickmanApp
: public Application
GameStack
GameStack : public
StateStack
State 3
State 2
Goes to
State 1
GameState: public
State
Has
Scene
A scene is composted of entities that
have components, and systems that act
upon those components.
Entities
Have
Components
Act on
Systems