광운대학교 컴퓨터정보공학부 특강

ECS 기반 게임 개발

옥찬호

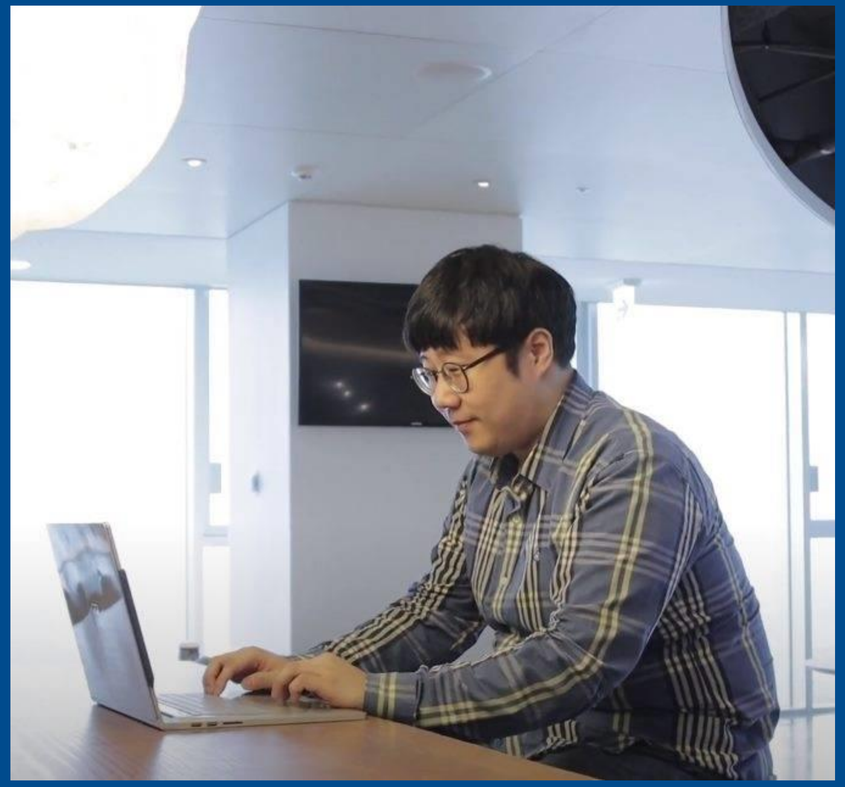
utilForever@gmail.com

발표자 소개

광운대학교 컴퓨터정보공학부 특강
ECS 기반 게임 개발

- 옥찬호 (Chris Ohk)
  - (현) Momenti Engine Engineer
  - (전) Nexon Korea Game Programmer
  - Microsoft Developer Technologies MVP
  - C++ Korea Founder & Administrator
  - Reinforcement Learning KR Administrator
  - IT 전문서 집필 및 번역 다수
    - 게임샐러드로 코드 한 줄 없이 게임 만들기 (2013)
    - 유니티 Shader와 Effect 제작 (2014)
    - 2D 게임 프로그래밍 (2014), 러스트 핵심 노트 (2017)
    - 모던 C++ 입문 (2017), C++ 최적화 (2019)

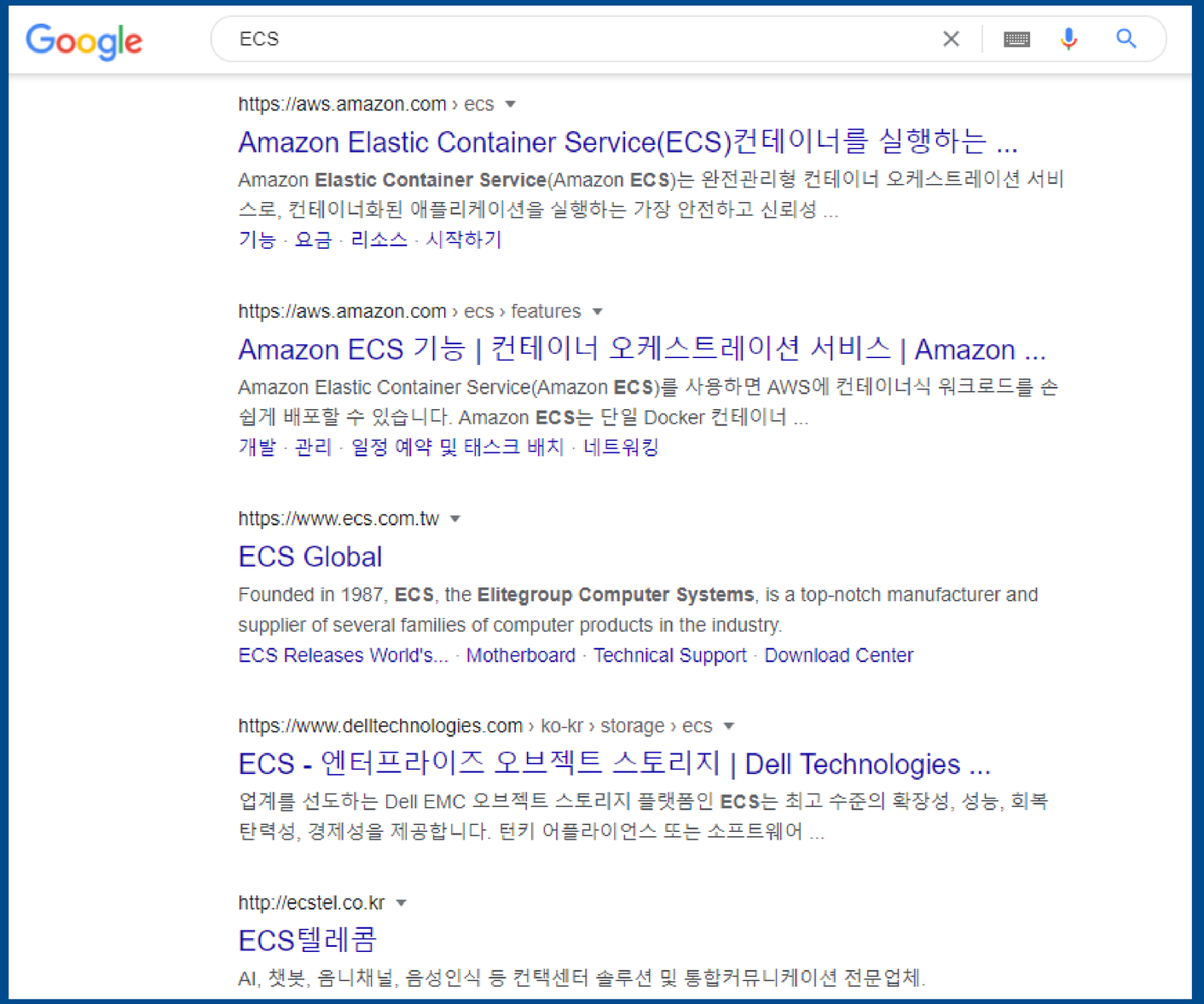
utilForever@gmail.com
utilForever



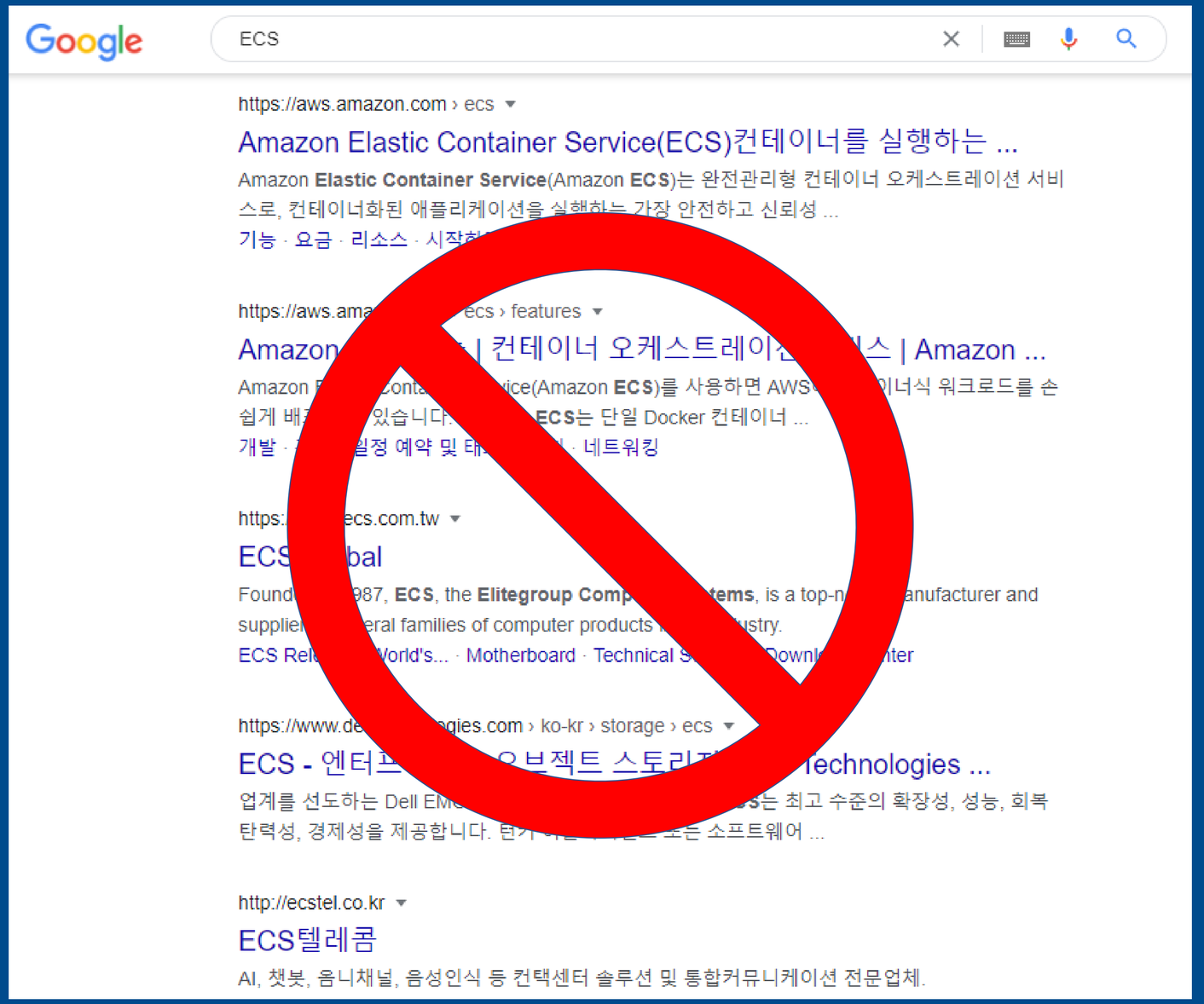
시작하기 전에

- 밋업의 발표 자료와 예제 코드는 다음 저장소에서 확인 가능합니다.
  https://github.com/utilForever/2022-KW-ECS-Development
- ECS를 사용한 코드에 관심이 있다면 다음 프로젝트를 참고하세요.
  - 타워 디펜스 : https://github.com/utilForever/CubbyTower
  - 하스스톤 전장 : https://github.com/utilForever/battlegrounds-rs

ECS


Google
ECS
https://aws.amazon.com › ecs
Amazon Elastic Container Service(ECS)컨테이너를 실행하는 ...
Amazon Elastic Container Service(Amazon ECS)는 완전관리형 컨테이너 오케스트레이션 서비스로, 컨테이너화된 애플리케이션을 실행하는 가장 안전하고 신뢰성 ...
기능 · 요금 · 리소스 · 시작하기
https://aws.amazon.com › ecs › features
Amazon ECS 기능 | 컨테이너 오케스트레이션 서비스 | Amazon ...
Amazon Elastic Container Service(Amazon ECS)를 사용하면 AWS에 컨테이너식 워크로드를 손쉽게 배포할 수 있습니다. Amazon ECS는 단일 Docker 컨테이너 ...
개발 · 관리 · 일정 예약 및 태스크 배치 · 네트워킹
https://www.ecs.com.tw
ECS Global
Founded in 1987, ECS, the Elitegroup Computer Systems, is a top-notch manufacturer and supplier of several families of computer products in the industry.
ECS Releases World's... · Motherboard · Technical Support · Download Center
https://www.delltechnologies.com › ko-kr › storage › ecs
ECS - 엔터프라이즈 오브젝트 스토리지 | Dell Technologies ...
업계를 선도하는 Dell EMC 오브젝트 스토리지 플랫폼인 ECS는 최고 수준의 확장성, 성능, 회복탄력성, 경제성을 제공합니다. 턴키 어플라이언스 또는 소프트웨어 ...
http://ecstel.co.kr
ECS텔레콤
AI, 챗봇, 옴니채널, 음성인식 등 컨택센터 솔루션 및 통합커뮤니케이션 전문업체.

ECS


Google
ECS
https://aws.amazon.com › ecs
Amazon Elastic Container Service(ECS)컨테이너를 실행하는 ...
Amazon Elastic Container Service(Amazon ECS)는 완전관리형 컨테이너 오케스트레이션 서비스로, 컨테이너화된 애플리케이션을 실행하는 가장 안전하고 신뢰성 ...
기능 · 요금 · 리소스 · 시작
https://aws.ama ecs › features
Amazon | 컨테이너 오케스트레이션 | Amazon ...
Amazon ice(Amazon ECS)를 사용하면 AWS 이너식 워크로드를 손쉽게 배 있습니다. ECS는 단일 Docker 컨테이너 ...
개발 · 일정 예약 및 태 · 네트워킹
https: ecs.com.tw
ECS bal
Found 987, ECS, the Elitegroup Comp tems, is a top-n anufacturer and supplie eral families of computer products ustry.
ECS Rel World's... · Motherboard · Technical S Downl nter
https://www.de gies.com › ko-kr › storage › ecs
ECS - 엔터프 오브젝트 스토리 Technologies ...
업계를 선도하는 Dell EM S는 최고 수준의 확장성, 성능, 회복 탄력성, 경제성을 제공합니다. 단 소프트웨어 ...
http://ecstel.co.kr
ECS텔레콤
AI, 챗봇, 음니채널, 음성인식 등 컨택센터 솔루션 및 통합커뮤니케이션 전문업체.

ECS

https://unity.com/kr/dots

ECS

- 게임 개발에서 주로 사용하는 소프트웨어 아키텍처 패턴 “Composition over Inheritance”의 원칙을 따름
- 상태와 행동(기능)을 분리
- 쿼리 기반 (캡슐화 없음, 메시지 전달 없음)

Entity **C**omponent **S**ystem

Entity

+

Component

+

System

Entity

- 컴포넌트들을 담아두는 곳
- 행동(기능)이나 데이터를 갖고 있지 않다.
- 어떤 데이터가 있는지 식별하는 역할을 한다.

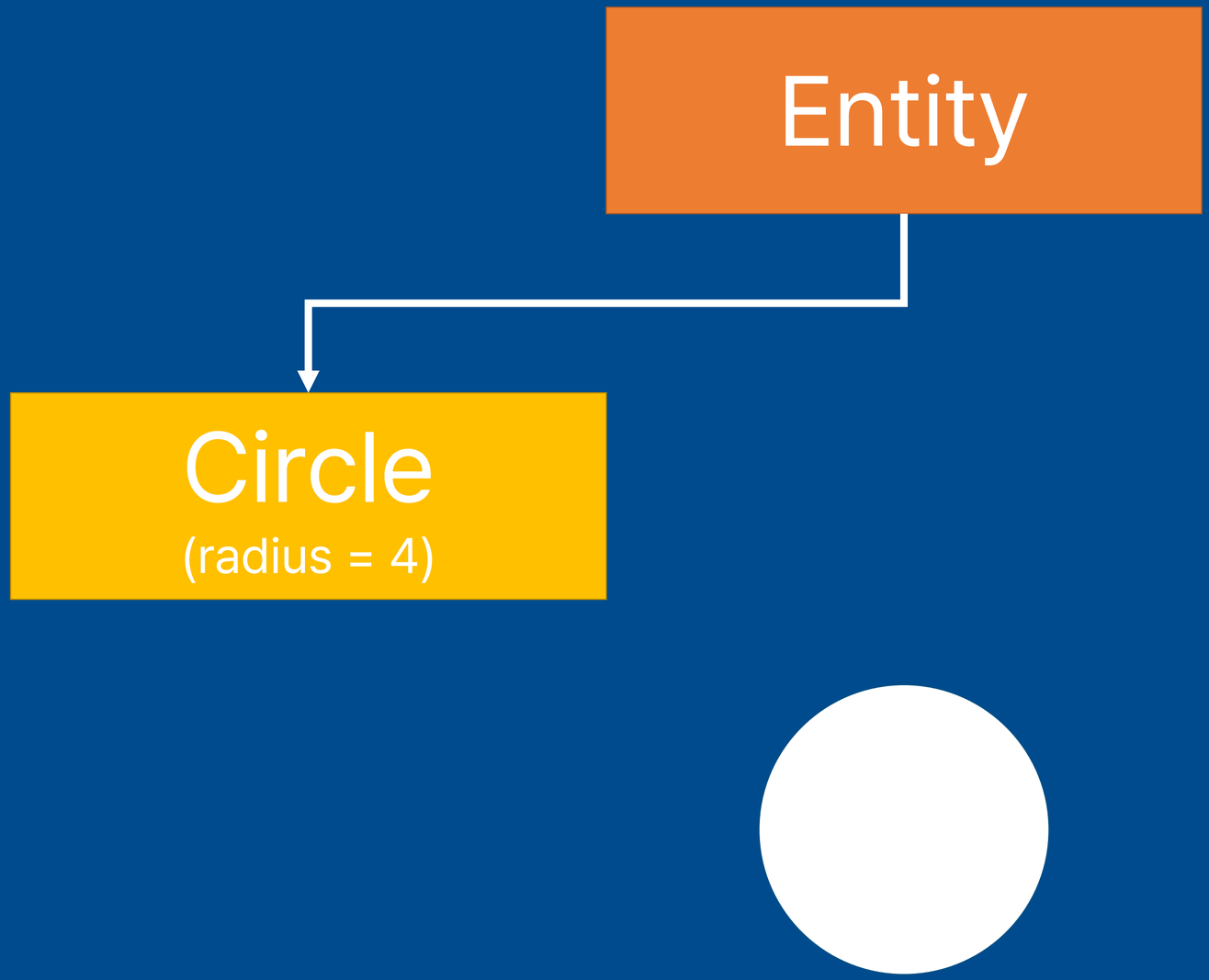
Component

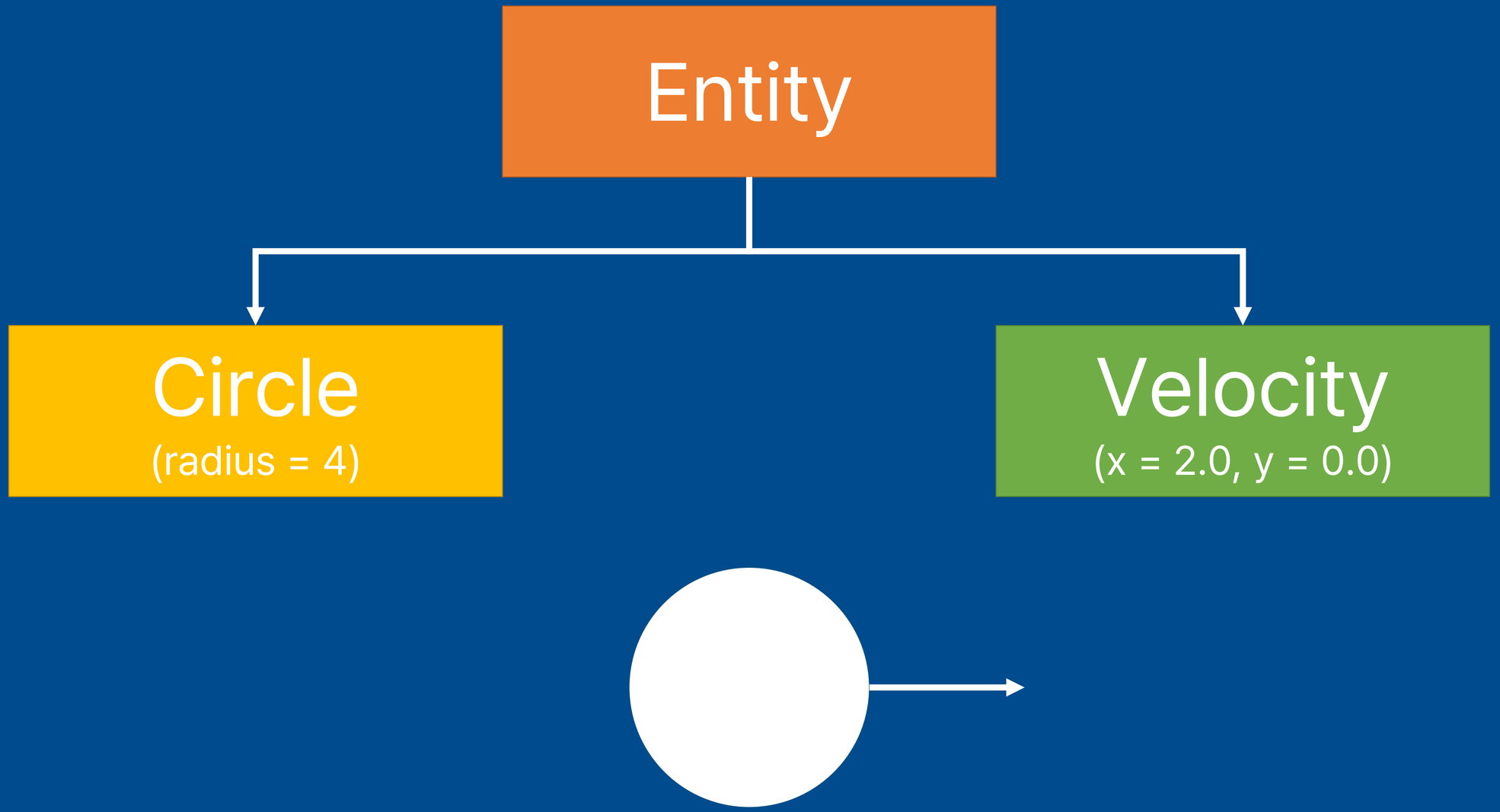
- 실제 데이터를 저장하는 곳
- 단순 태그일 수도 있고, 값을 저장하는 컨테이너일 수도 있고,
  무언가를 참조하는 컨테이너일 수도 있다.

System

- 컴포넌트의 데이터를 현재 상태에서 다음 상태로 변환하는 논리
- 전체 엔티티 중 특정 컴포넌트를 가리키는 엔티티만 필터링한 다음, 컴포넌트와 관련해 필요한 동작을 수행한다.
  - 데이터를 변경하거나
  - 새로운 컴포넌트를 추가하거나
  - 기존 컨테이너를 삭제하거나

Entity


Entity
Circle
(radius = 4)

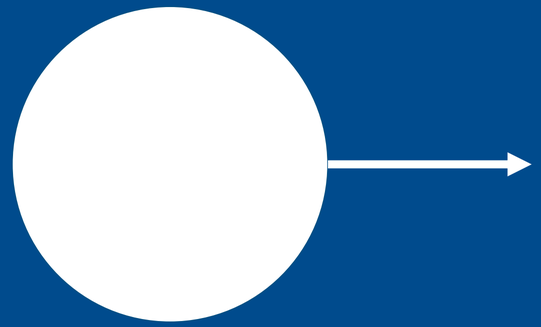

Entity
Circle
(radius = 4)
Velocity
(x = 2.0, y = 0.0)

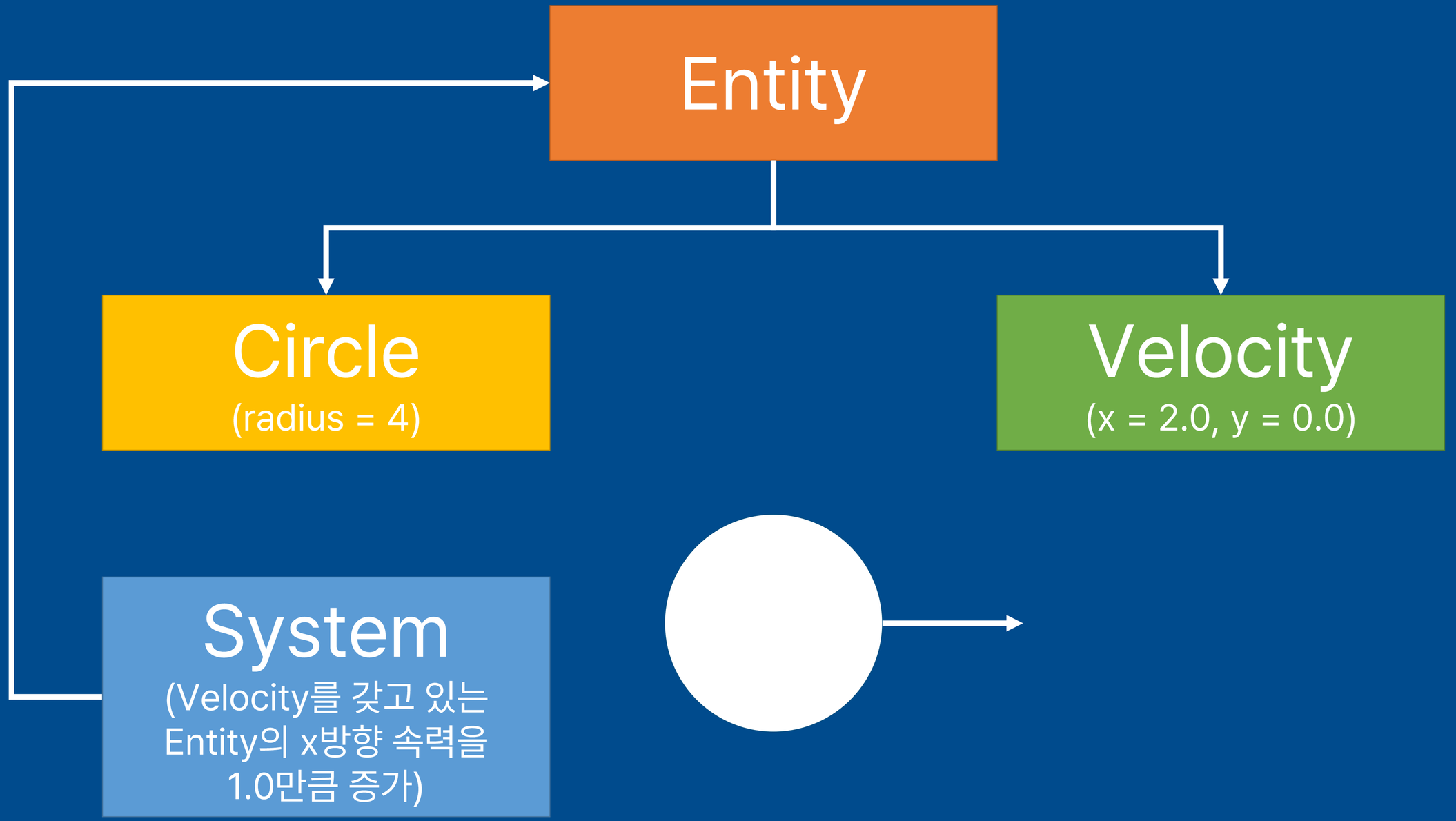
Entity

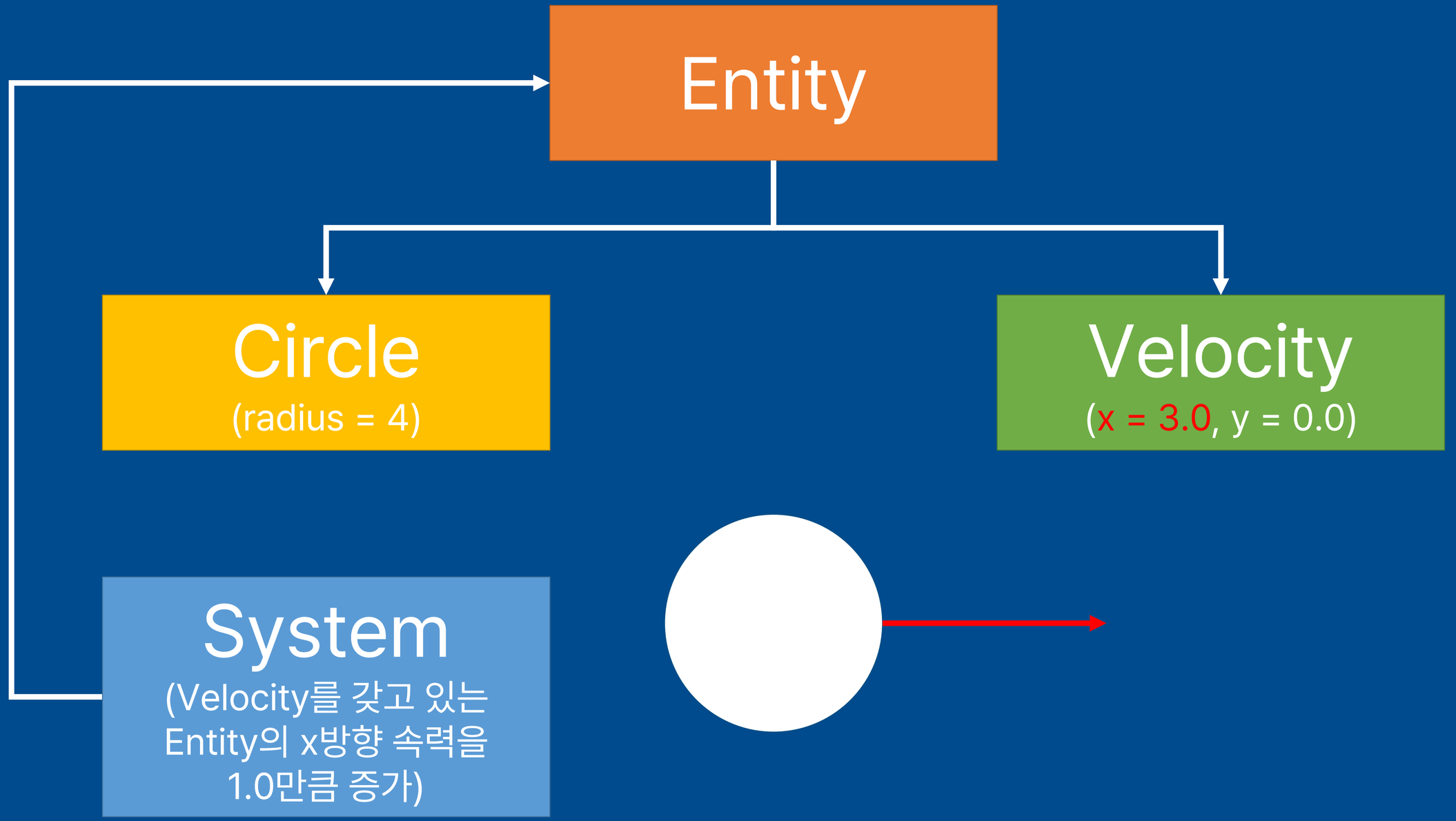
Circle
(radius = 4)

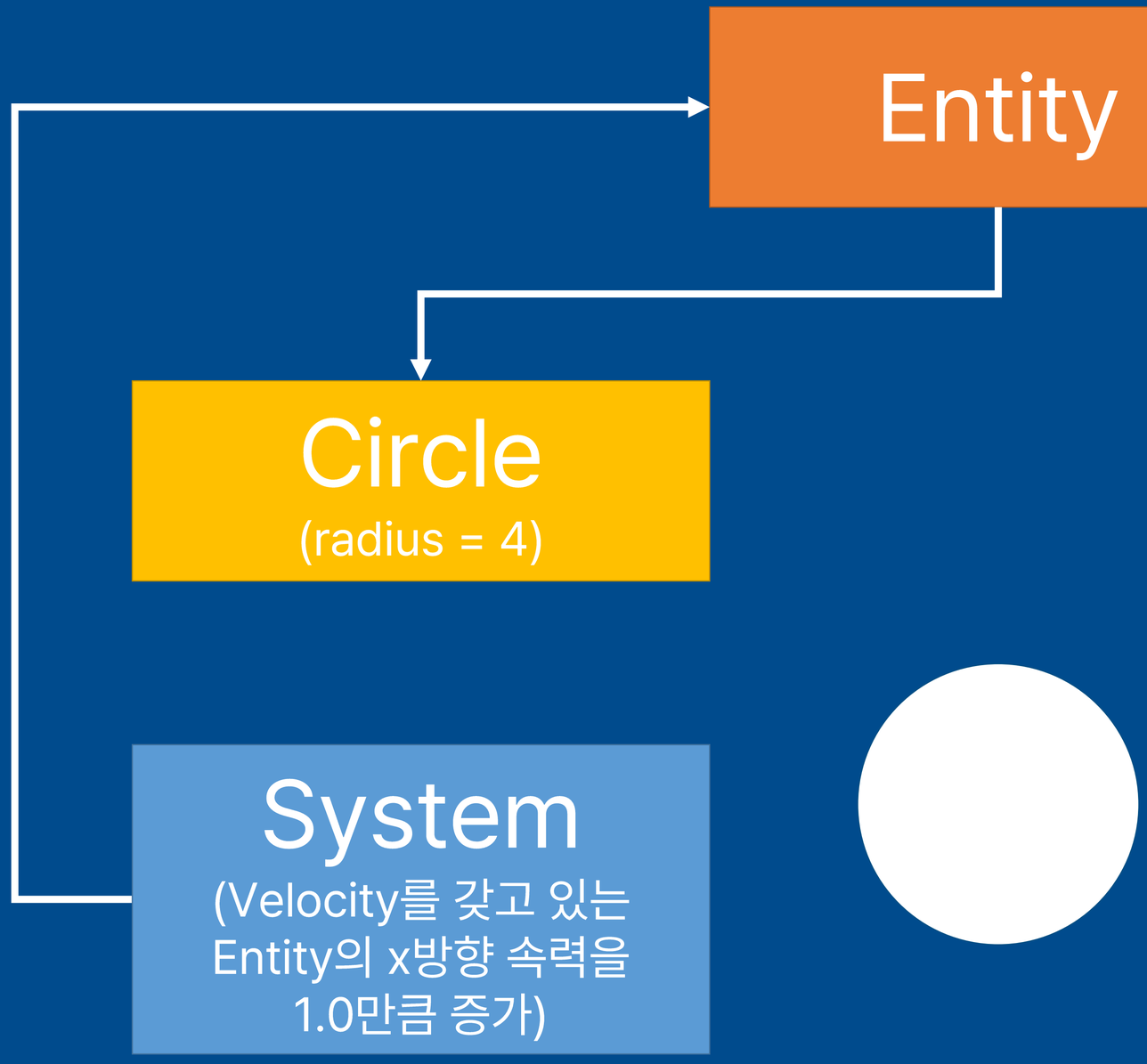
Velocity
(x = 2.0, y = 0.0)

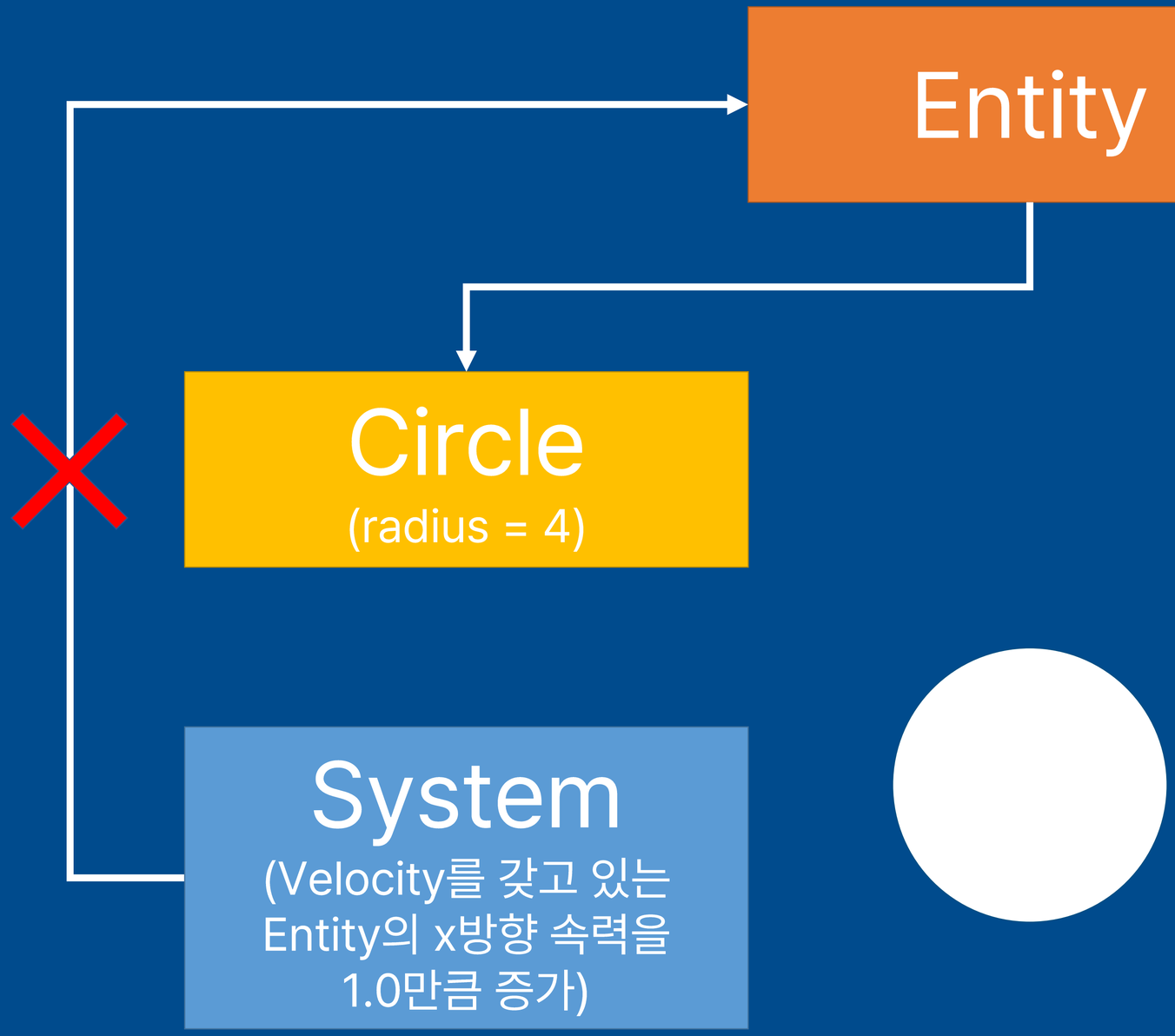
System
(Velocity를 갖고 있는
Entity의 x방향 속력을
1.0만큼 증가)

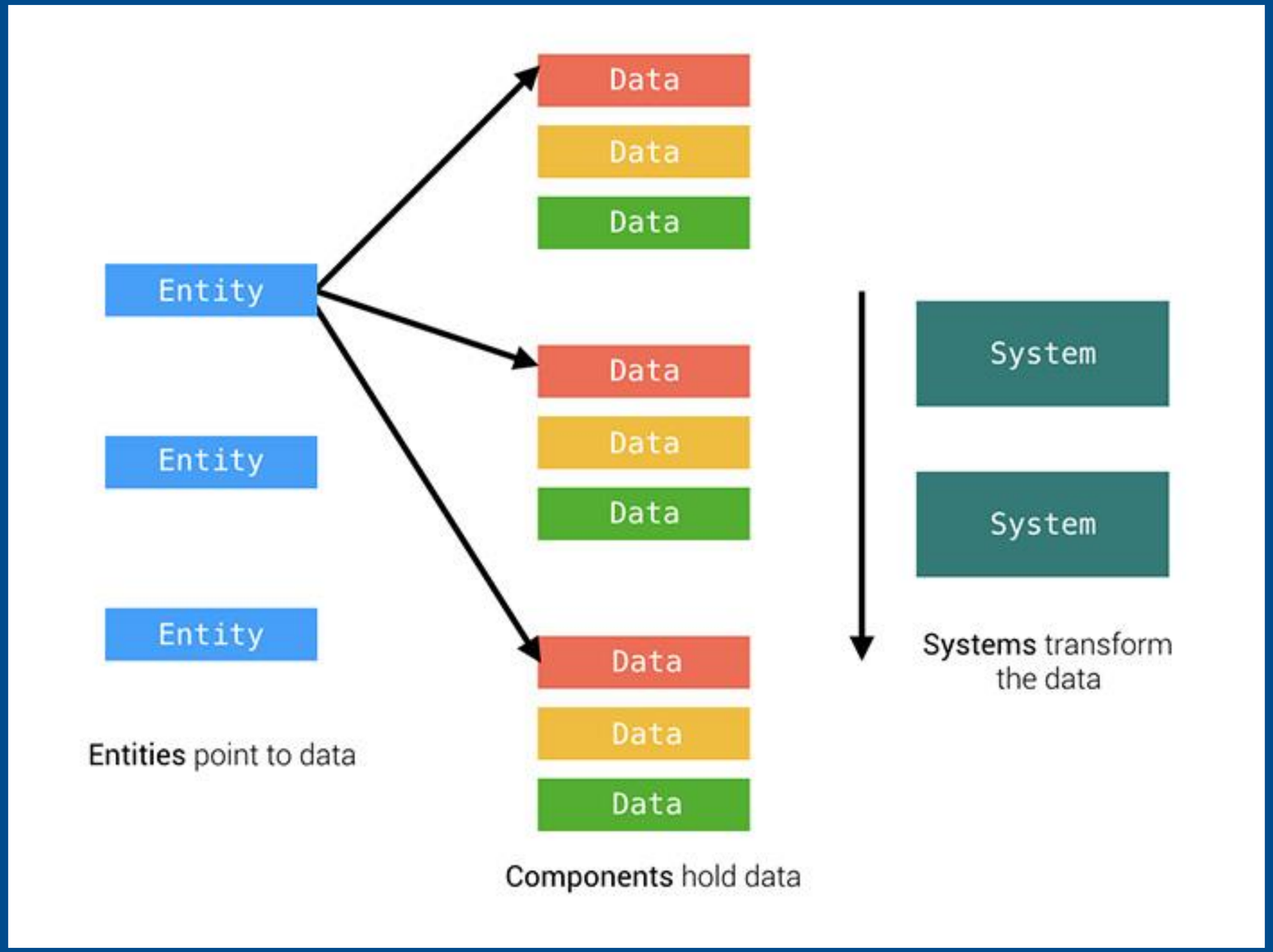



Entity
Circle
(radius = 4)
Velocity
(x = 2.0, y = 0.0)
System
(Velocity를 갖고 있는
Entity의 x방향 속력을
1.0만큼 증가)


Entity
Circle
(radius = 4)
Velocity
(x = 3.0, y = 0.0)
System
(Velocity를 갖고 있는
Entity의 x방향 속력을
1.0만큼 증가)


Entity
Circle
(radius = 4)
System
(Velocity를 갖고 있는
Entity의 x방향 속력을
1.0만큼 증가)


Entity
Circle
(radius = 4)
System
(Velocity를 갖고 있는
Entity의 x방향 속력을
1.0만큼 증가)



https://www.raywenderlich.com/7630142-entity-component-system-for-unity-getting-started

ECS를 쓰는 이유

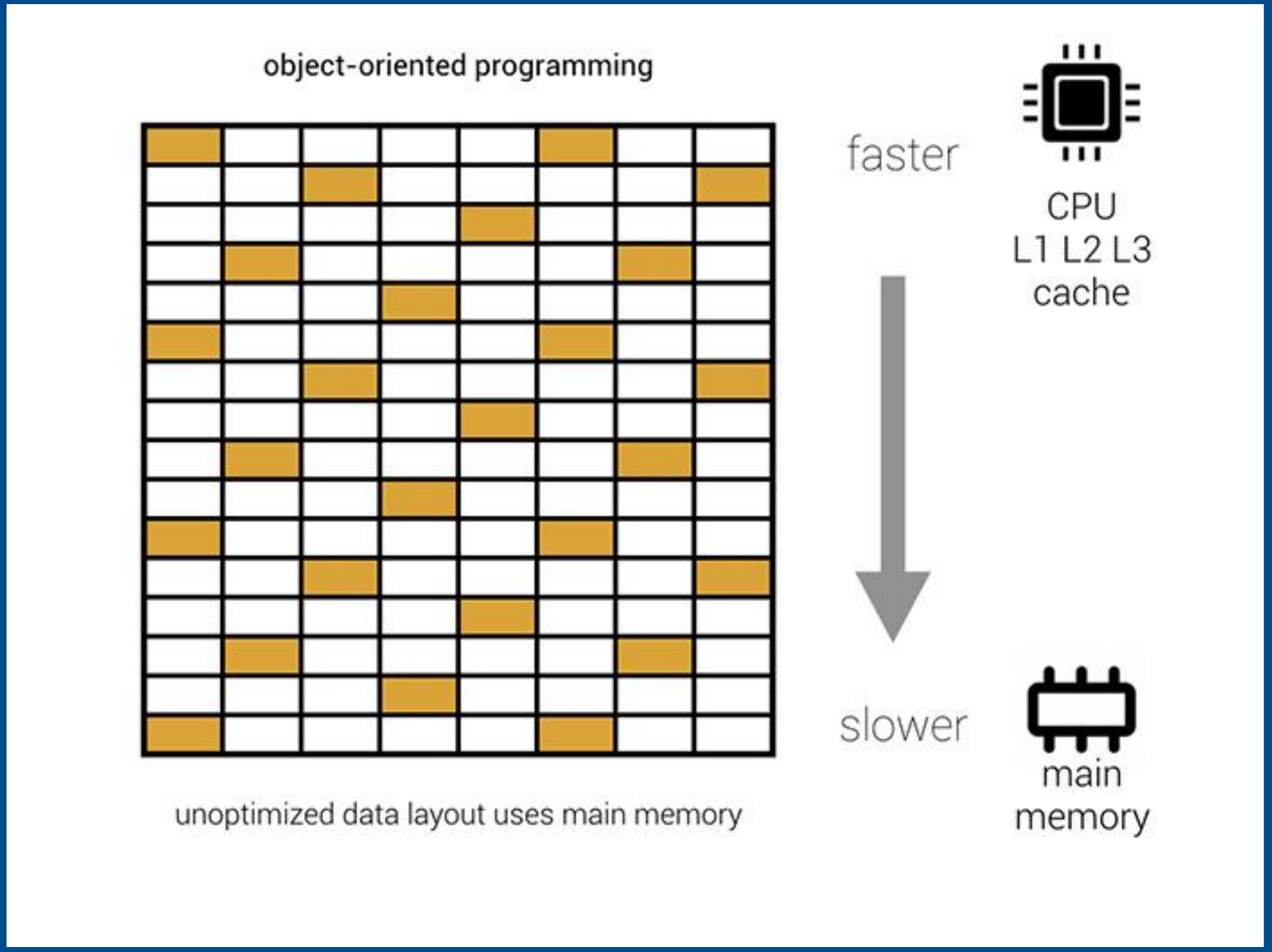
- 객체 지향 프로그래밍에 비해 성능이 뛰어나다.
- 그 이유는 데이터 지향 디자인(Data Oriented Design) 때문이다.
- 지역성(Locality) 개념을 알고 있다면 이해하기 쉽다.

지역성

- 캐시가 효율적으로 동작하려면, 캐시에 저장할 데이터가 지역성을 가져야 한다.
- 지역성이란 데이터 접근이 시간적, 혹은 공간적으로 가깝게 일어나는 것을 의미한다.
- 공간적 지역성
  - 특정 데이터와 가까운 주소가 순서대로 접근되었을 경우를 공간적 지역성이라고 한다.
    CPU 캐시나 디스크 캐시의 경우 한 메모리 주소에 접근할 때 그 주소뿐 아니라
    해당 블록을 전부 캐시에 가져오게 된다.
    이때 메모리 주소를 오름차순이나 내림차순으로 접근한다면, 캐시에 이미 저장된
    같은 블록의 데이터를 접근하게 되므로 캐시의 효율성이 크게 향상된다.

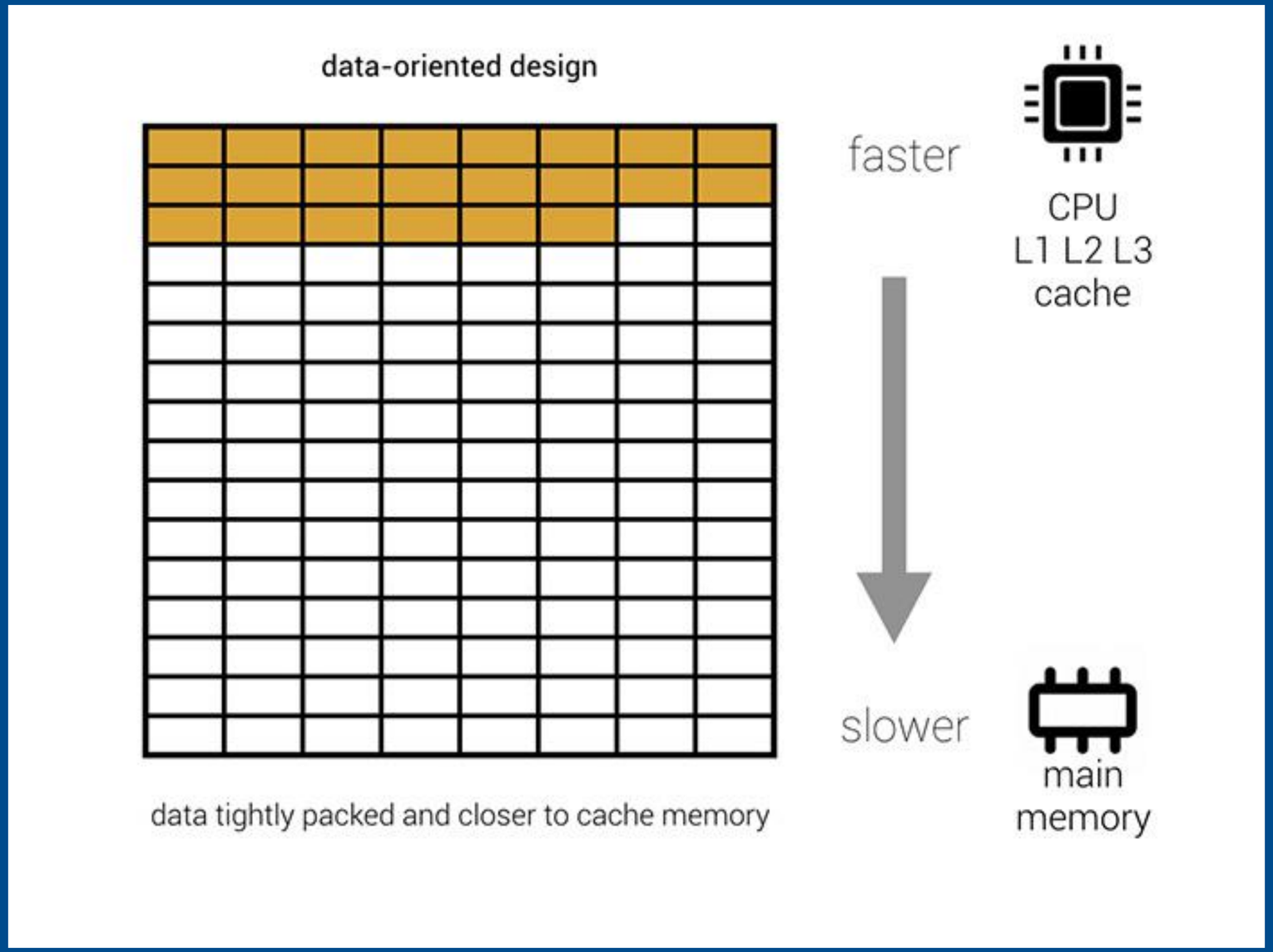
```cpp
class Circle
{
public:
    Circle() = default;
    Circle(std::string name, float radius) : m_name(name), m_radius(radius)
    {
        m_position = std::make_pair(0.0f, 0.0f);
        m_velocity = std::make_pair(0.0f, 0.0f);
    }

    void IncreaseSpeed(float xVel, float yVel)
    {
        m_velocity.first += xVel;
        m_velocity.second += yVel;
    }

private:
    std::string m_name;
    float m_radius;
    std::pair<float, float> m_position;
    std::pair<float, float> m_velocity;
};
```

```cpp
std::vector<Circle> circles;
circles.reserve(1000);

for (int i = 0; i < 1000; ++i)
{
    circles.emplace_back(Circle("Circle", 4.0f))
}

for (auto& circle : circles)
{
    circle.IncreaseSpeed(1.0f, 1.0f);
}
```



https://www.raywenderlich.com/7630142-entity-component-system-for-unity-getting-started

```cpp
std::vector<std::string> names;
std::vector<std::float> radii;
std::vector<std::pair<float, float>> positions;
std::vector<std::pair<float, float>> velocities;

...
velocities.reserve(1000);

for (int i = 0; i < 1000; ++i)
{
    ...
    velocities.emplace_back(std::make_pair(0.0f, 0.0f));
}

for (auto& velocity : velocities)
{
    velocity.first += 1.0f;
    velocity.second += 1.0f;
}
```



https://www.raywenderlich.com/7630142-entity-component-system-for-unity-getting-started

ECS 라이브러리

- C++ : entt, EntityX, Fflecs
- C# : DefaultEcs, Svelto.ECS
- Python : esper
- Rust : Specs, Legion
- Lua : tiny-ecs
- JavaScript : bitECS, ESCY

entt

- https://github.com/skypjack/entt
- 헤더 파일 하나를 포함해 사용 가능
- 마인크래프트 및 마인크래프트 어스에서 사용

entt


```cpp
#include <entt/entt.hpp>

struct position {
    float x;
    float y;
};

struct velocity {
    float dx;
    float dy;
};
```

entt


```cpp
int main() {
    entt::registry registry;

    for(auto i = 0u; i < 10u; ++i) {
        const auto entity = registry.create();
        registry.emplace<position>(entity, i * 1.f, i * 1.f);
        if(i % 2 == 0) { registry.emplace<velocity>(entity, i * .1f, i * .1f); }
    }

    update(registry);
}
```

entt


```cpp
void update(entt::registry &registry) {
    auto view = registry.view<const position, velocity>();

    // use a callback
    view.each([](const auto &pos, auto &vel) { /* ... */ });

    // use an extended callback
    view.each([](const auto entity, const auto &pos, auto &vel) { /* ... */ });

    // use a range-for
    for(auto [entity, pos, vel]: view.each()) {
        // ...
    }

    // use forward iterators and get only the components of interest
    for(auto entity: view) {
        auto &vel = view.get<velocity>(entity);
        // ...
    }
}
```

게임 개발에 ECS 적용해 보기

- 간단한 타워 디펜스 게임을 만든다고 생각해 보자.
- 플레이어는 골드를 지불하고 타워를 건설할 수 있다.
  또한 타워를 업그레이드 할 수 있다.
- 보유하고 있는 골드가 타워 건설 비용보다 적으면 건설할 수 없다.
  또한 업그레이드 비용보다 적으면 업그레이드할 수 없다.
- 이를 entt를 사용해 구현해 보자.

게임 개발에 ECS 적용해 보기


```cpp
void Initialize(entt::registry& registry)
{
    // Player
    {
        auto entity = registry.create();
        registry.emplace<Tag::Player>(entity);
        registry.emplace<Gold>(entity, 500);
    }
}
```

게임 개발에 ECS 적용해 보기


```
void BuyArrowTower(entt::registry& registry)
{
    // Check the player can buy arrow tower
    if (!Withdraw(registry, registry.view<Tag::Player>()[0],
                  ARROW_TOWER_LV1_PRICE))
    {
        return;
    }

    auto entity = registry.create();
    registry.emplace<Tag::Tower>(entity);
    registry.emplace<Name>(entity, "Arrow Tower Lv 1");
    registry.emplace<Upgradable>(entity, ARROW_TOWER_LV2_PRICE,
                                 UpgradeArrowTowerLv2);
}
```

게임 개발에 ECS 적용해 보기


```cpp
bool Withdraw(entt::registry& registry, entt::entity from, int amount)
{
    if (!registry.all_of<Gold>(from))
    {
        return false;
    }

    if (auto& gold = registry.get<Gold>(from); gold.amount >= amount)
    {
        gold.amount -= amount;
        return true;
    }

    return false;
}
```

게임 개발에 ECS 적용해 보기


```cpp
void UpgradeArrowTowerLv2(entt::registry& registry, entt::entity entity)
{
    registry.replace<Name>(entity, "Arrow Tower Lv 2");
    registry.remove<Upgradable>(entity);
}
```

게임 개발에 ECS 적용해 보기


```
void UpdateUpgradeSystem(entt::registry& registry, entt::entity from)
{
    if (registry.all_of<Upgradable>(from))
    {
        if (const auto& upgradable = registry.get<Upgradable>(from); Withdraw(
                registry, registry.view<Tag::Player>()[0], upgradable.cost))
        {
            upgradable.Upgrade(registry, from);
        }
    }
}
```

게임 개발에 ECS 적용해 보기


```
//!
//! \brief Upgradable struct.
//!
//! This struct stores the cost and the function to upgrade something.
//!
struct Upgradable
{
    int cost;
    std::function<void(entt::registry&, entt::entity)> Upgrade;
};
```

정리

- ECS는 데이터 지향 디자인을 통해 성능에 초점을 맞춘 패턴이다.
- Entity는 데이터를 가리키며, Component는 데이터를 저장하며, System은 데이터를 변환한다.
- OOP로 작성된 코드를 ECS로 바꾸려면 생각의 전환이 필요하다.
- 하지만 ECS로 바꾸고 나면 게임 로직이 좀 더 직관적으로 보인다.
- 기존과 다른 새로운 방식으로 개발해보고 싶다면 추천!

참고 자료

- https://en.wikipedia.org/wiki/Entity_component_system
- https://www.raywenderlich.com/7630142-entity-component-system-for-unity-getting-started
- https://mrbinggrae.tistory.com/223
- https://youtu.be/DJx0JKuemvA

감사합니다.

utilForever@gmail.com

https://github.com/utilForever

Facebook, Twitter: @utilForever