GAME PROGRAMMING IN C++

2020

Main loop

```
// Main loop
SDL_Event event{ };
while (m_Running)
{
    while (SDL_PollEvent(&event) != 0)
    {
        if (event.type == SDL_QUIT)
        {
            m_Running = false;
        }
    }

    float argumentForUpdate = 1.0f; // TODO: Remove
    Update(argumentForUpdate);
}
```

Input Manager

```cpp
void InputManager::ProcessInput()
{
    for (auto& [action, key] : m_InputActions)
    {
        bool bIsPressed = KeyDown(key);
        switch (m_InputActionStates[action])
        {
        case EInputActionState::None:
        {
            m_InputActionStates[action] = bIsPressed ? EInputActionState::JustPressed : EInputActionState::None;
            break;
        }
        case EInputActionState::JustPressed:
        case EInputActionState::Pressed:
        {
            m_InputActionStates[action] = bIsPressed ? EInputActionState::Pressed : EInputActionState::Released;
            break;
        }
        case EInputActionState::Released:
        {
            m_InputActionStates[action] = bIsPressed ? EInputActionState::JustPressed : EInputActionState::None;
            break;
        }
        default:
            ASSERT("Unknown EInputActionState {0}", m_InputActionStates[action]);
            m_InputActionStates[action] = EInputActionState::None;
            break;
        }
    }
}
```

Precompiled Headers

- Header
  - Ne kompajlira se direktno
  - #include-uje se u druge fajlove
- Veliki broj čestih includeova povećava trajanje kompilacije
- Spojiti i isprocesiratete include-ova kako bi se ubrzala kompilacija
- https://blog.knatten.org/2010/10/29/what-i-have-against-precompiled-headers/

Linearn algebra u C++

- Ne postoji u standardu
- Linear algebra proposal: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1385r1.html - Možda u C++23
- U međuvremenu, biblioteke
  - glm, eigen, hlsl++, uBLAS, etl, blaze, Armadillo

GLM

- OpenGL Mathematics (GLM) is a header only C++ mathematics library for graphics software based on the OpenGL Shading Language (GLSL) specifications.

- GLM provides classes and functions designed and implemented with the same naming conventions and functionalities than GLSL so that anyone who knows GLSL, can use GLM as well in C++.

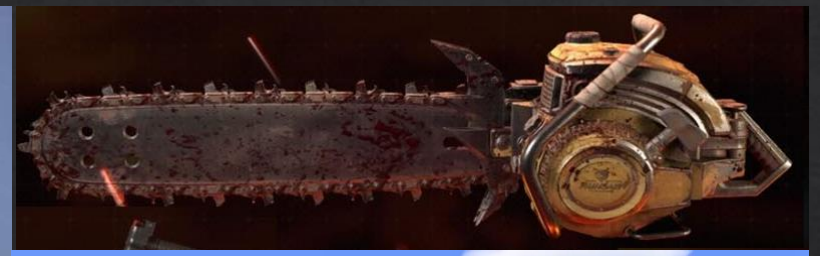
- Used by by including the `<glm/glm.hpp>` header

Entity Component System

- Šta je Entitet?



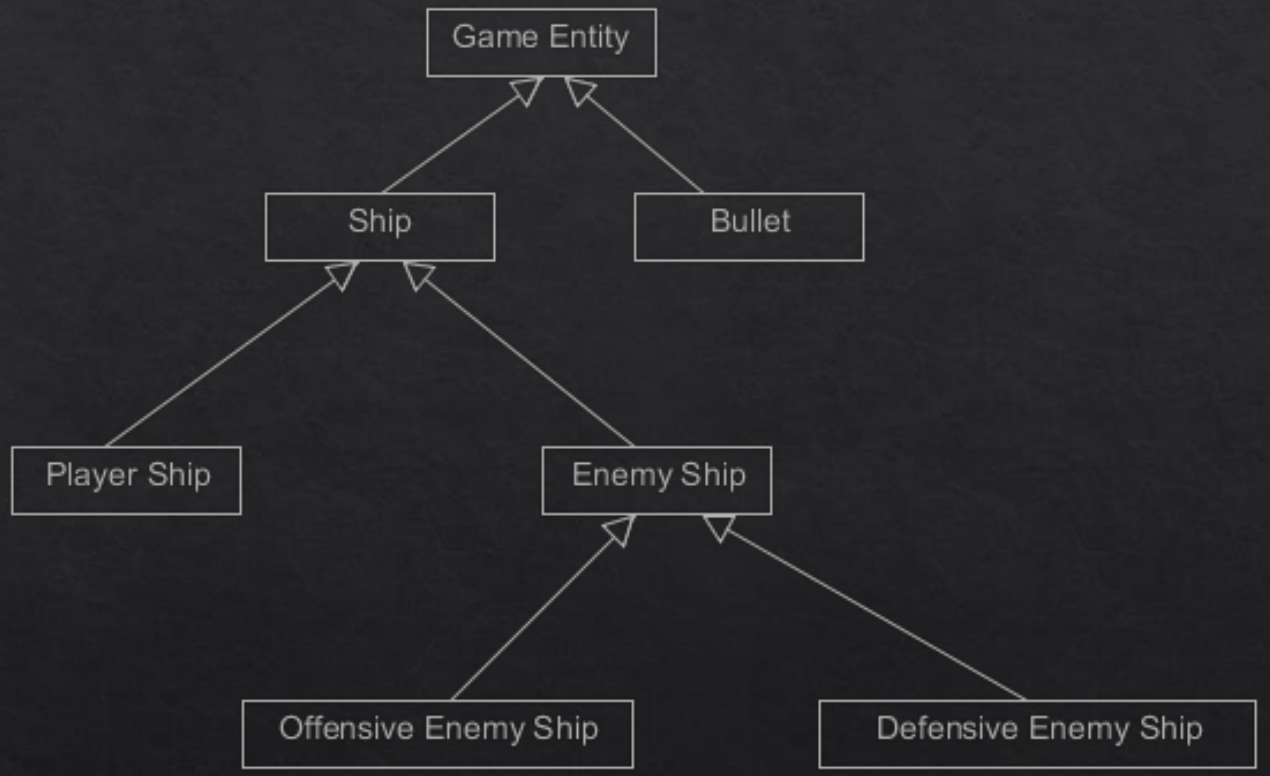


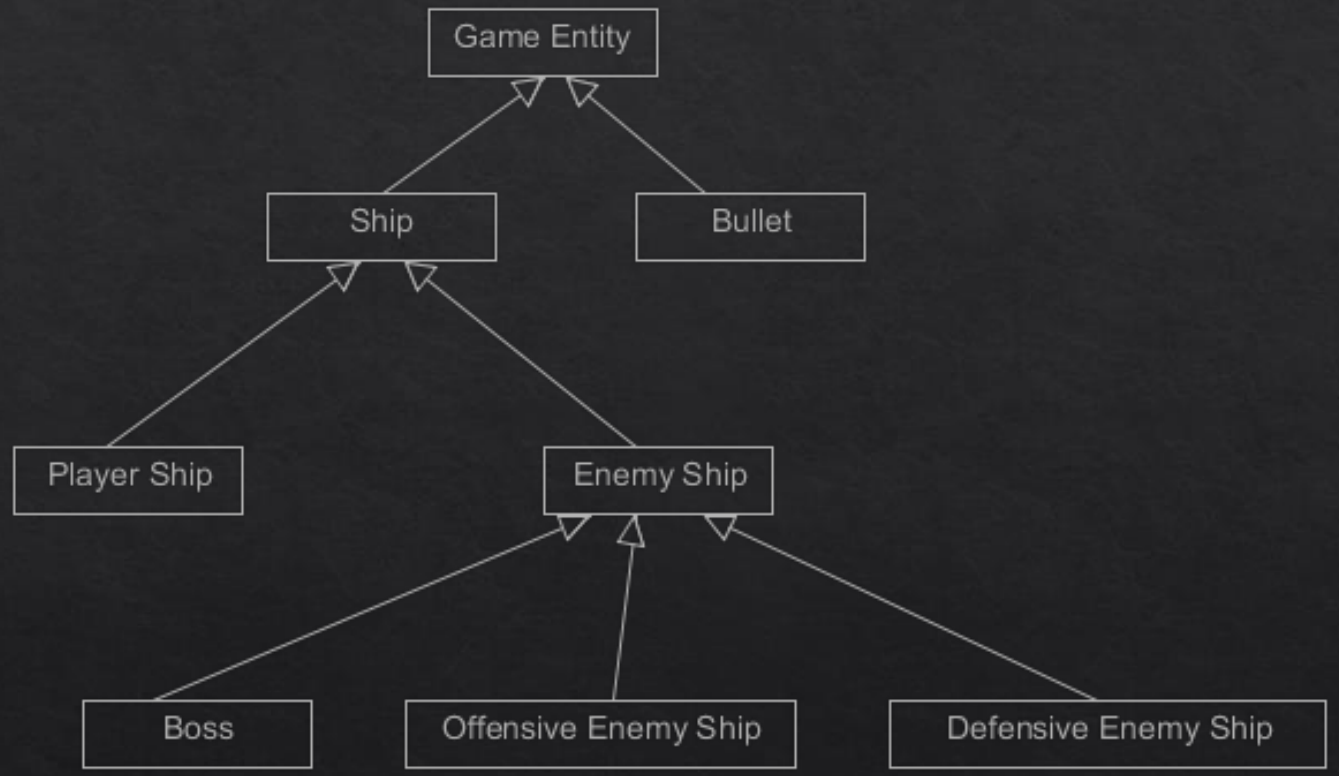

Šta je entitet?

- Objekat u svetu
- Može (a ne mora) biti:
  - vidljiv
  - pokretljiv
  - agresivan
  - eksplodirajuć
  - naciljan
  - klikabilan (?)
  - pametan
  - …….
- Pojam entiteta se pojavljuje u svim žanrovima igara

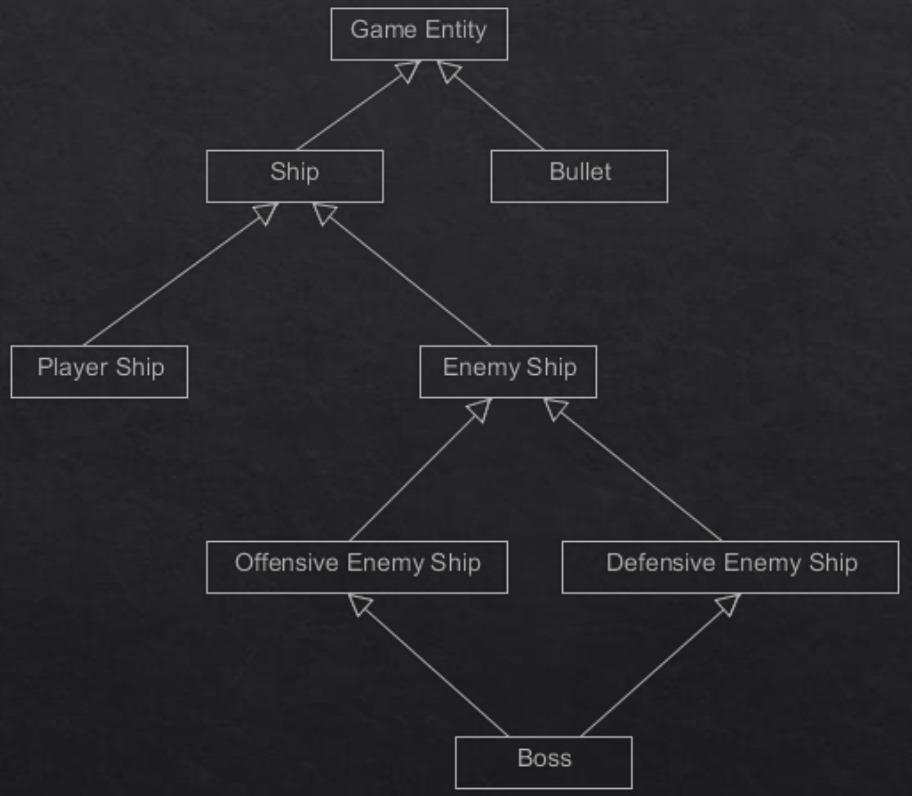
OOP arhitektura entiteta



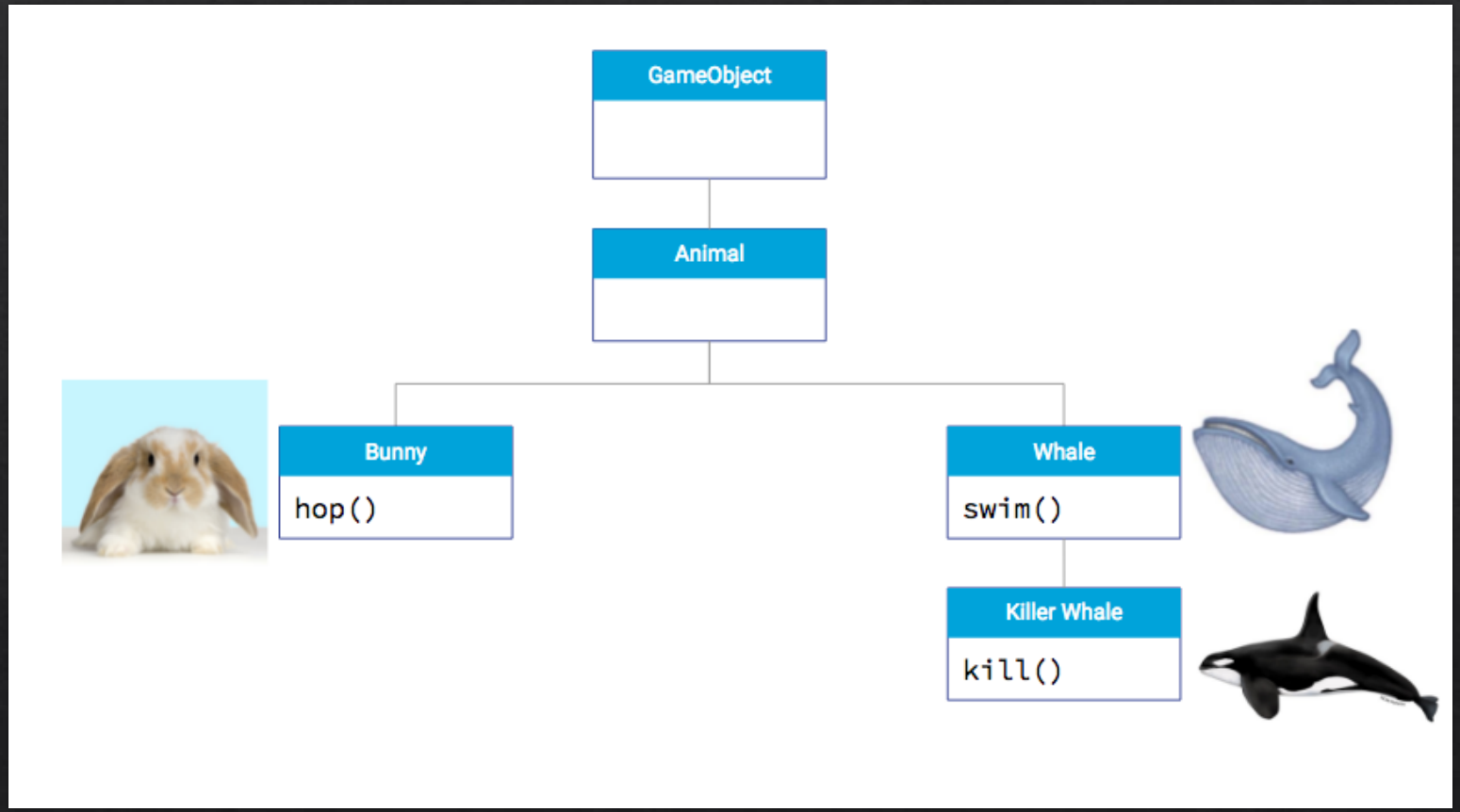
OOP arhitektura entiteta



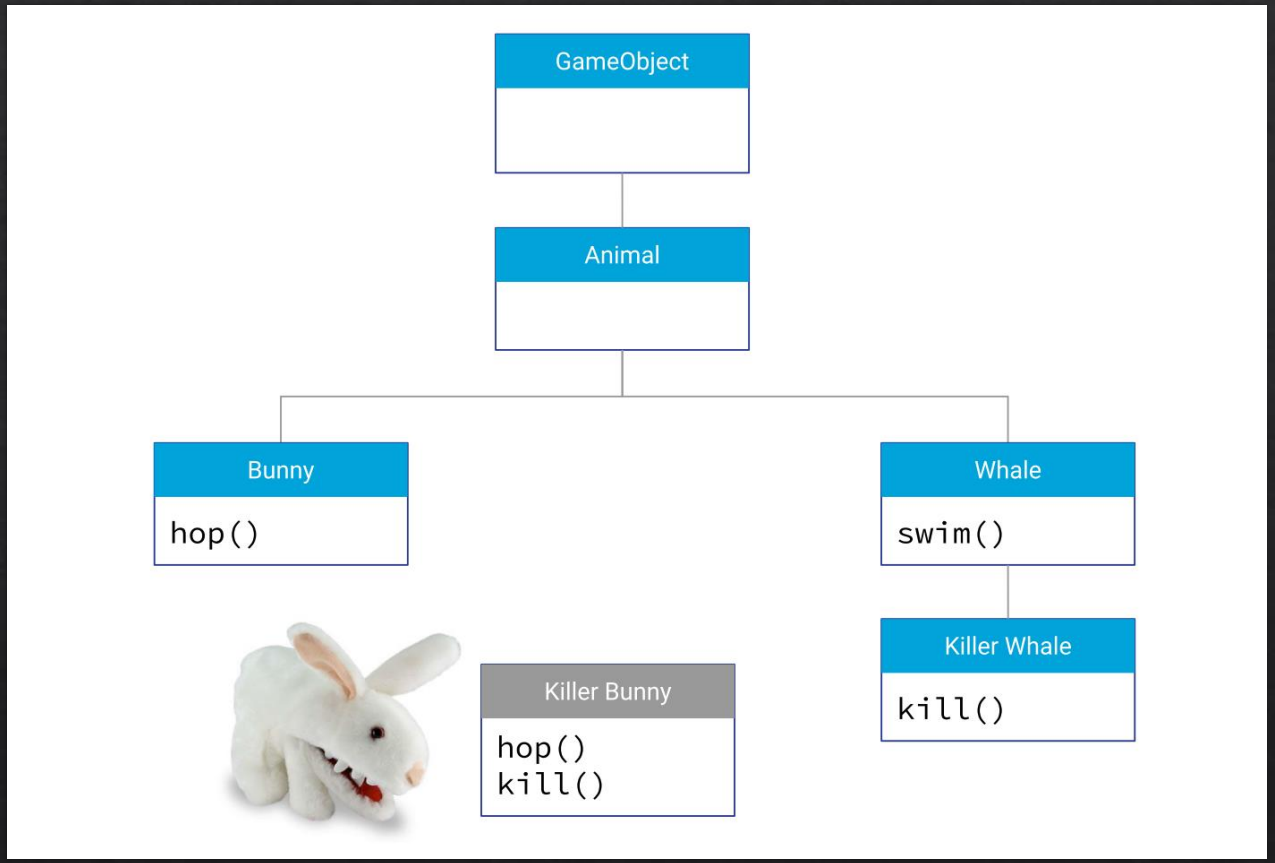
OOP arhitektura entiteta



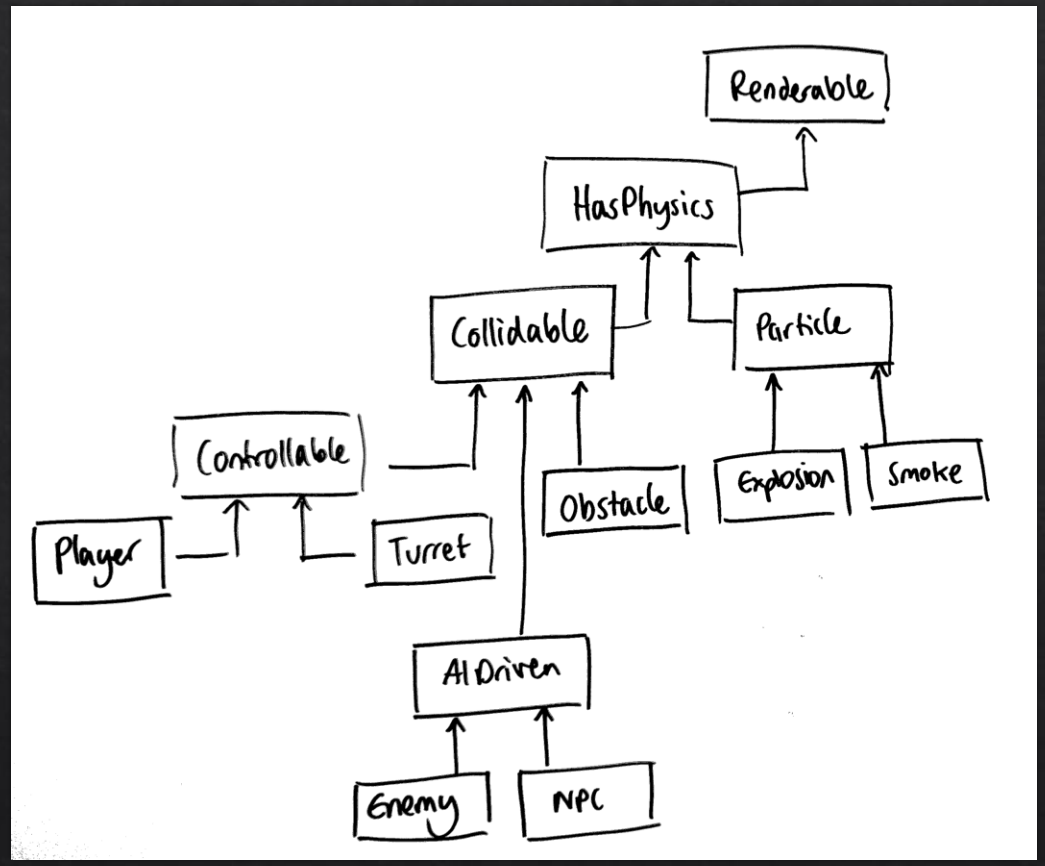
OOP arhitektura entiteta



OOP arhitektura entiteta



OOP arhitektura entiteta



OOP arhitektura entiteta

- Entitet je bazna klasa
  - Klasa Entitet i klase koje je nasleđuju sadrže osnovnu logiku igre i podatke o stanju igre
- Velike klase
  - Često dodajemo virtuelne funkcije (prazne!) u baznu klasu samo da bismo mogli da podržimo funkcionalnost koja nama treba u izvedenim klasama
  - Kod koji dodajemo u najniže izvedene klase često moramo da kopiramo da bismo izbegli diamond inheritance
- Mnogi sistemi ne pripadaju jednom objektu
  - Kolizije, AI…

OOP arhitektura entiteta

- Duboke i rigidne hijerarhije
- Moramo da poznajemo sve klase iz kojih nasleđujemo
- Nemoguće je osigurati se da se metode bazne klase pozivaju iz nasleđene klase
  - Provešćete celo veče debagujući jedan `base.Update()` koji fali negde.
- Mnogo virtuelnih poziva
- Teško za multithreading

OOP arhitektura entiteta

- http://whats-in-a-game.com/implementation-inheritance-is-evil/
- https://channel9.msdn.com/Events/GoingNative/2013/Inheritance-Is-The-Base-Class-of-Evil
- https://www.gamedev.net/blogs/entry/2265481-oop-is-dead-long-live-oop/

The purpose of all programs, and all parts of those programs, is to transform data from one form to another.”

“If you don’t understand the data, you don’t understand the problem.”

— Mike Acton

"Data-Oriented Design and C++”, CppCon 2014, https://www.youtube.com/watch?v=rX0ItVEVjHc

bmcnett
@bmcnett

lot of people who'd never drive to the store to buy one slice of bread, see nothing strange in driving out to RAM to read one integer.

5:54 AM - 22 Sep 2018

When there is One, there is Many

```
virtual void Update(double time, float deltaTime) override
{
    /* move one thing */
}
```

```
void UpdateAllMoves(size_t n, GameObject* objects, double time, float deltaTime)
{
    /* move all of them */
}
```

http://aras-p.info/texts/files/2018Academy%20-%20ECS-DoD.pdf

“Object composition over class inheritance”

- Klasa Entity - Ne postoje podaci i funkcionalnosti u Entitetu
- Klasa Component, bazna klasa za sve komponente
- Entitet sadrži niz komponenti koje zapravo sadrže podatke i funkcionalnost

- Lakše za razumevanje i širenje
- Lako dodavanje novih koncepata bez kršenja postojećeg koda

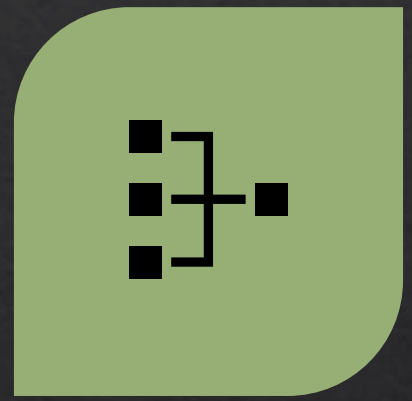
- Entity-Component arhitektura

- Šta je sa sistemima?

ECS

ENTITY

COMPONENT



SYSTEM

Možemo još bolje

- Ne postoje podaci i funkcionalnosti u Entitetu
- Bez metoda u komponentama, one samo sadrže podatke
- Sva funkcionalnost se nalazi **u Sistemima**
  - PhysicsSystem
  - HealthSystem
  - FightSystem
  - RenderSystem

- Sistemi operišu nad komponentama

Osobine ECS

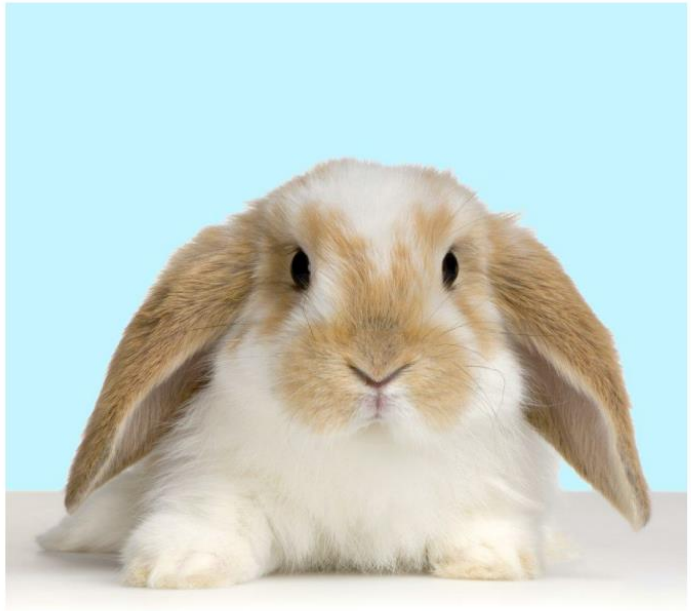
- Razdvajanje ponašanja i podataka u odvojene klase
- Možemo da dodajemo i brišemo komponente u runtime-u
- Svaka komponenta i sistem je zasebna jedinica koju možemo zasebno da testiramo
- Lako za proširivanje – samo dodamo novi tip komponente i novi tip sistem
- Jednostavnije za paralelizaciju

Osobine ECS

- Kako rešiti komunikaciju između različitih komponenti?
- Kako rešiti zavisnost između sistema?

Primeri

The bunny entity



What components might a bunny have?

Placeable
- *int* x
- *int* y
- *int* z

Seeing
- *int* sight_radius
- *boolean* night_vision?

Huggable
- *int* fluffiness

Living
- *float* health
- *float* age

Consumable
- *float* calories

Hopping
- *int* hop_distance

Physical
- *int* height
- *int* width
- *int* length

The carrot entity

Placeable
- *int* x
- *int* y
- *int* z

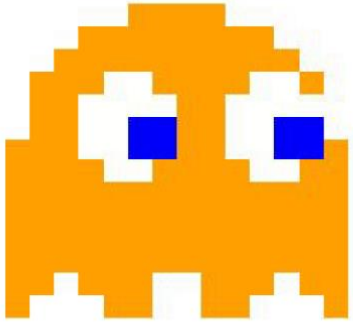
Consumable
- *float* calories

Physical
- *int* height
- *int* width
- *int* length

The ghost entity

Placeable
- *int* x
- *int* y
- *int* z

Seeing
- *int* sight_radius
- *boolean* night_vision?



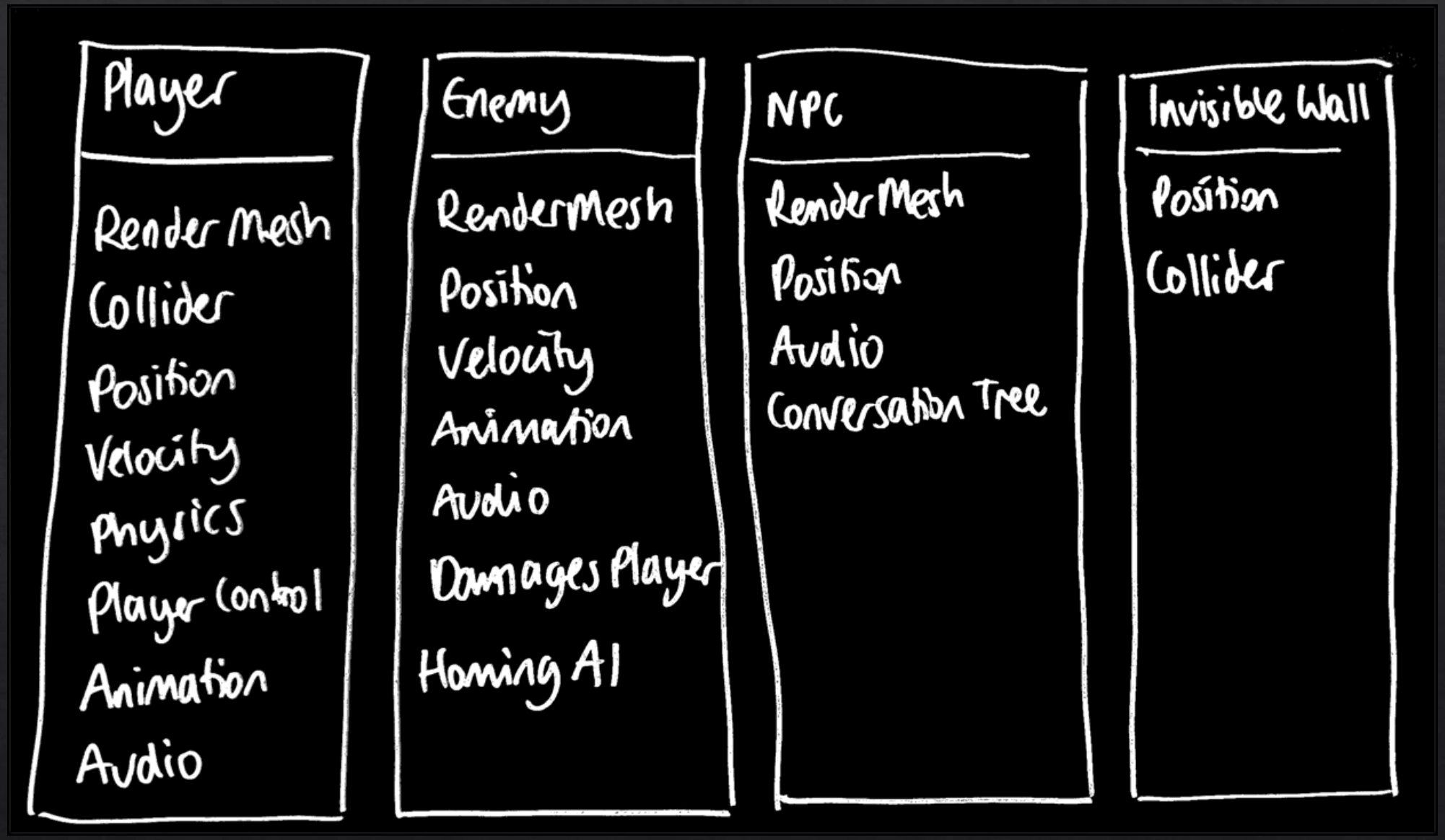
Spooky
- *int* spookiness

Komponente

- Kvalitativni opis Entiteta
- Mali objekti koji se sastoje samo od podataka (POD)
- Nemaju ponašanje

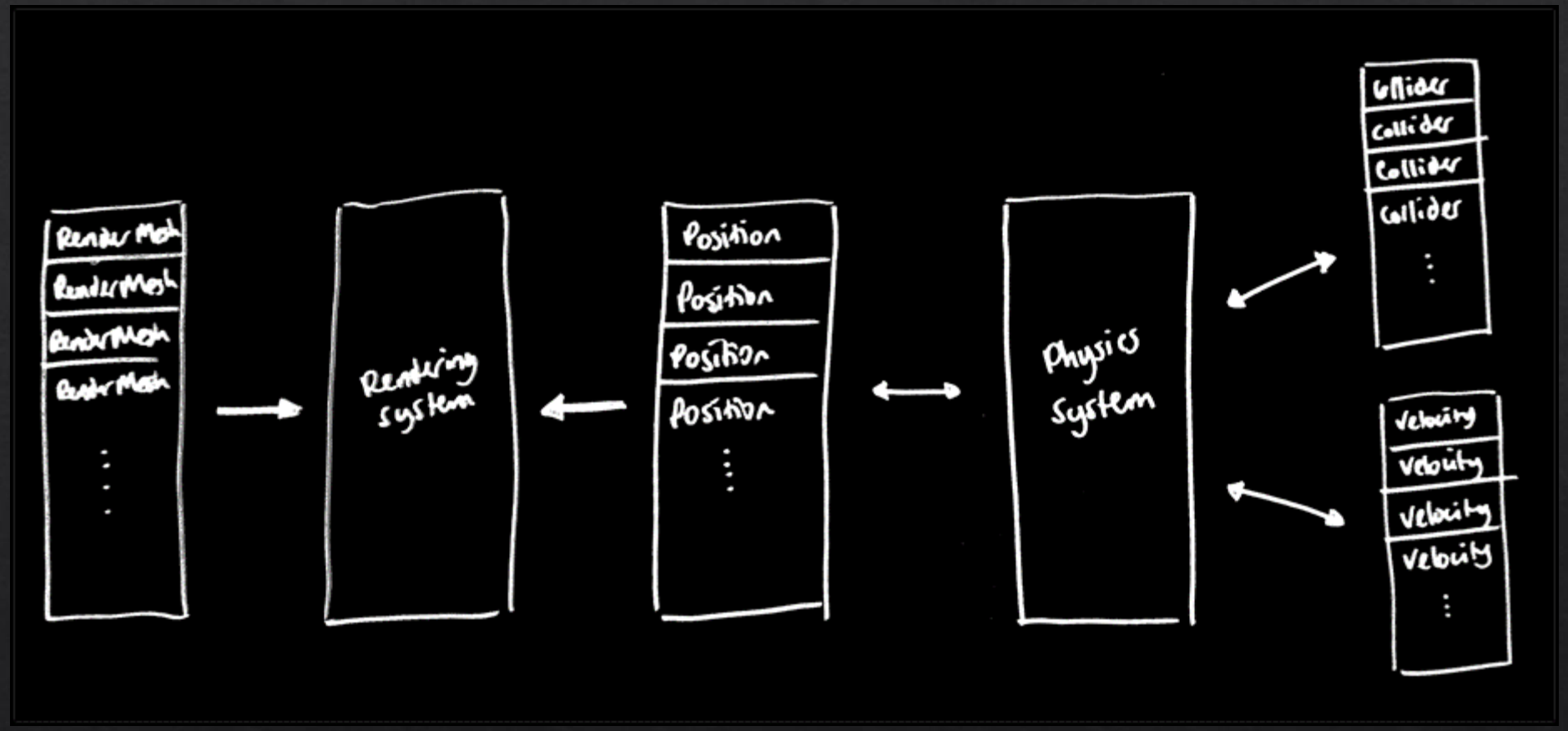
Entitet

- Skup komponenti
- Ne sadrže direktno podatke
- Ne sadrže ponašanje
- Predstavljaju zbir svojih komponenti


Player
RenderMesh
Collider
Position
Velocity
Physics
Player Control
Animation
Audio
Enemy
RenderMesh
Position
Velocity
Animation
Audio
Damages Player
Homing AI
NPC
RenderMesh
Position
Audio
Conversation Tree
Invisible Wall
Position
Collider

Sistem

- Daju život entitetima i komponentama
- Neprekidno iteriraju kroz sve komponente određenog tipa (ili određenih tipova)
- Sistemi čitaju vrednosti iz komponenata i upisuju vrednosti u komponente


RenderMesh
RenderMesh
RenderMesh
RenderMesh
Rendering system
Position
Position
Position
Position
Physics System
Collider
Collider
Collider
Velocity
Velocity
Velocity
Velocity

How does a bunny behave?

Placeable
- x 5
- y -2
- z 10

"Fall"

Gravity System

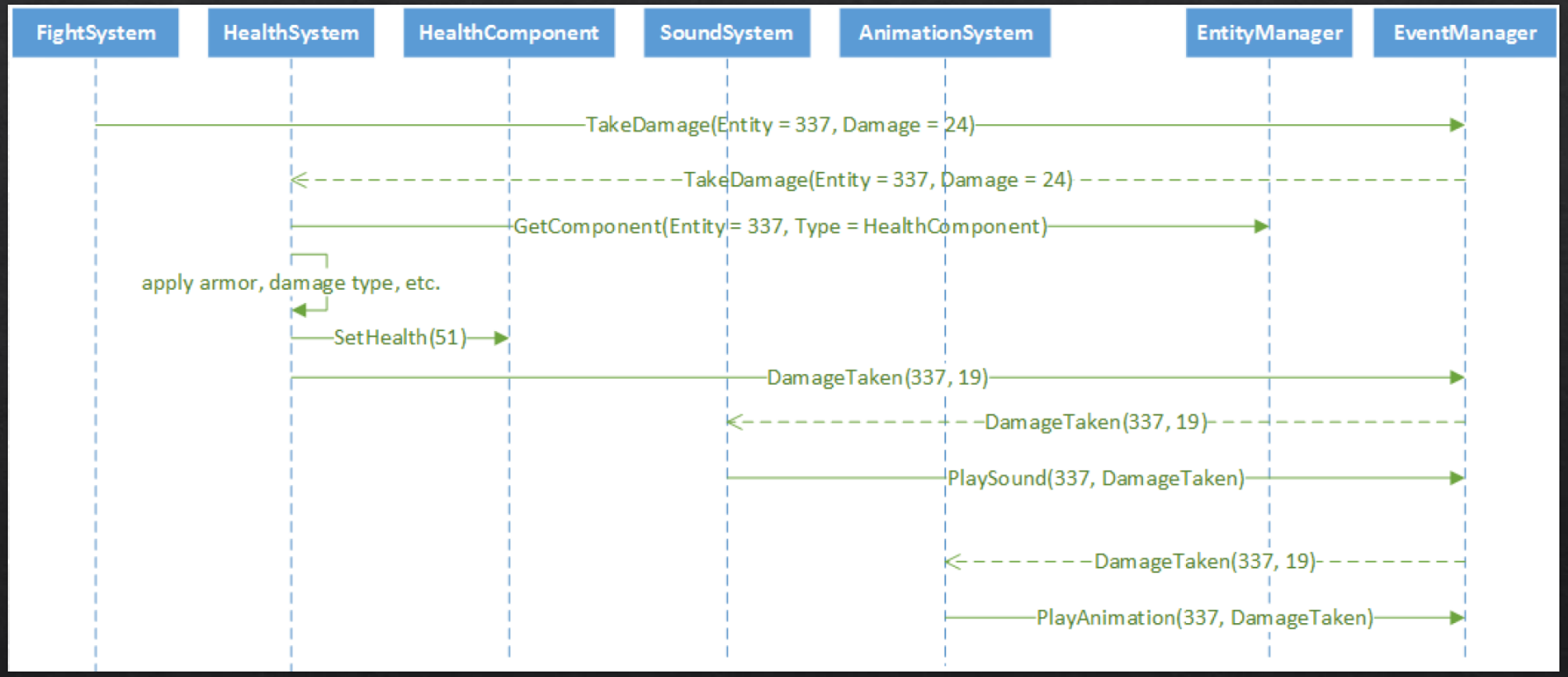
Placeable
- x 5
- y -2
- z 0

Living
- age 2.00

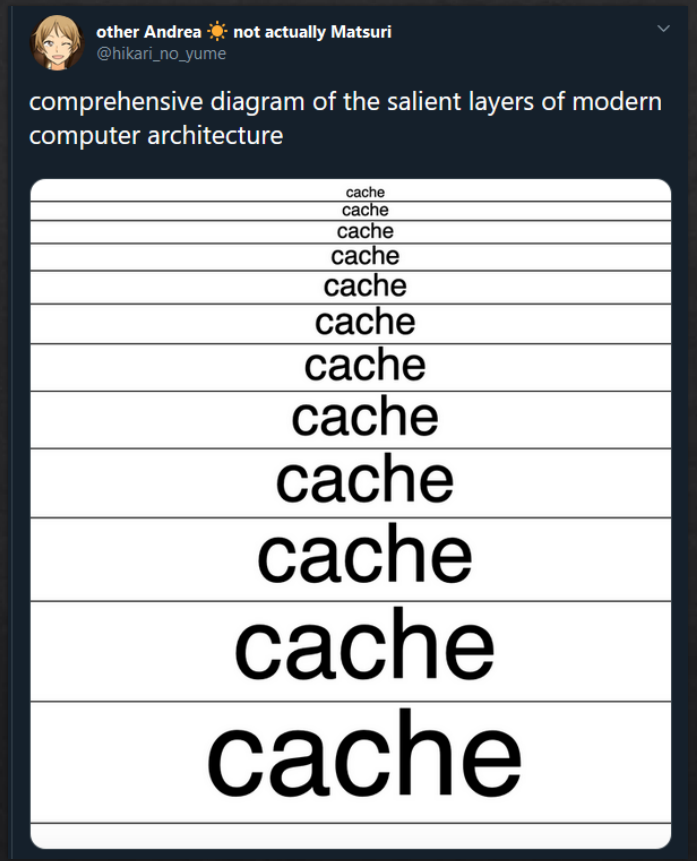
Time System

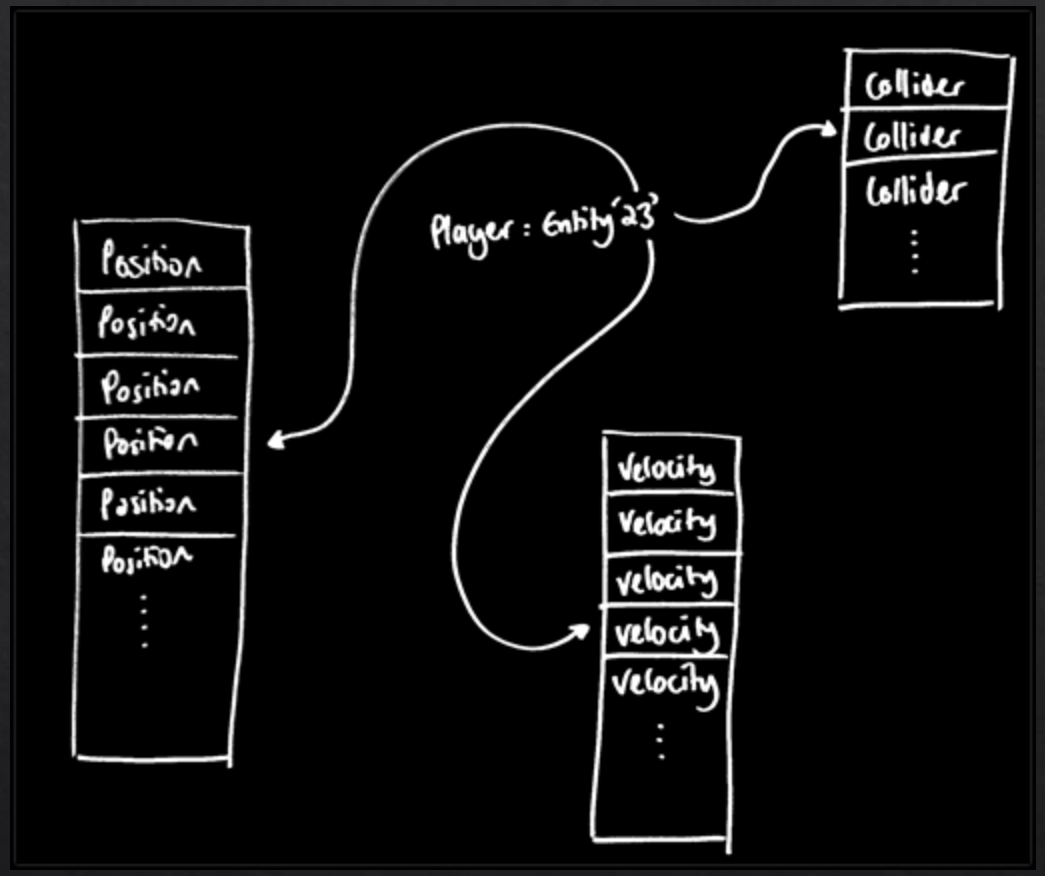
"Age"

Living
- age 2.01


FightSystem
HealthSystem
HealthComponent
SoundSystem
AnimationSystem
EntityManager
EventManager
TakeDamage(Entity = 337, Damage = 24)
TakeDamage(Entity = 337, Damage = 24)
GetComponent(Entity = 337, Type = HealthComponent)
apply armor, damage type, etc.
SetHealth(51)
DamageTaken(337, 19)
DamageTaken(337, 19)
PlaySound(337, DamageTaken)
DamageTaken(337, 19)
PlayAnimation(337, DamageTaken)

Da li možemo još bolje?




Collider
Collider
Collider
Player : Entity '23'
Position
Position
Position
Position
Position
Position
Velocity
Velocity
Velocity
Velocity
Velocity

Literatura

- Data-Oriented Design and C++", CppCon 2014
- Overwatch Gameplay Architecture and Netcode, GDC
- https://yos.io/2016/09/17/entity-component-systems/
- http://bitsquid.blogspot.com/2014/08/building-data-oriented-entity-system.html
- https://medium.com/ingeniouslysimple/entities-components-and-systems-89c31464240d
- Entity Component System Overview in 7 Minutes
- The Entity-Component-System - An awesome game-design pattern in C++ (Part 1)
- https://www.slideshare.net/npruehs/game-programming-02-componentbased-entity-systems?from_action=save
- CppCon 2015: Vittorio Romeo "Implementation of a component-based entity system in modern C++"
- Understanding data-oriented design for entity component systems - Unity at GDC 2019
- Amethyst Specs - Parallel ECS written in Rust

Naš ECS

Component

```cpp
struct Component
{
    Entity* m_Owner{};
    ComponentTypeID m_TypeId{ 0 };
    bool m_Active{ false };
    virtual ~Component() = default;

private:
    inline static ComponentTypeID m_MaxComponentTypeId = 0;

public:
    template <typename T>
    static ComponentTypeID GetComponentTypeID()
    {
        static_assert (std::is_base_of<Component, T>::value, "");
        static ComponentTypeID typeID = Component::m_MaxComponentTypeId++;
        return typeID;
    }
};
```

Entity

```
class Entity
{
public:
    unsigned int GetId() const { return m_Id; }

protected:
    inline static unsigned int m_CurrentId = 0;
    unsigned int m_Id;
    std::vector<std::unique_ptr<Component>> m_Components;

public:
    Entity() { m_Id = m_CurrentId++; }
```

Entity::AddComponent

```cpp
template <typename TComponent, typename... TArgs>
TComponent& AddComponent(TArgs&&... mArgs)
{
    if (HasComponent<TComponent>())
    {
        ASSERT(false, "Attempting to add a component twice! Entity ID: {}, ComponentType: {}", m_Id, Component::GetComponentTypeID<TComponent>());
    }

    auto component = std::make_unique<TComponent>(std::forward<TArgs>(mArgs)...);

    component->m_TypeId = Component::GetComponentTypeID<TComponent>();

    m_Components.push_back(std::move(component));

    return *(static_cast<TComponent*>(m_Components.back().get()));
}
```

```cpp
Entity* e = new Entity;
e->AddComponent<TransformComponent>(200.f, 200.f);
e->AddComponent<CollisionComponent>(200.f, 200.f);
e->AddComponent<SpriteComponent>();
```

EntityManager

```
class EntityManager
{
public:
    bool Init();
    void Update(float dt);
    void AddEntity(Entity* e);
    void AddEntity(std::unique_ptr<Entity>&& e);
```

EntityManager::GetAllEntitiesWithComponents

```cpp
template <typename... TComponent>
auto GetAllEntitiesWithComponents()
{
    std::vector<Entity*> returnVec{};

    for (const auto& entity : m_Entities)
    {
        if (entity->HasComponents<TComponent...>())
        {
            returnVec.push_back(entity.get());
        }
    }

    return returnVec;
}
```

```cpp
template <typename... TComponent>
bool HasComponents() const
{
    if ( (HasComponent<TComponent>() && ...) )
    {
        return true;
    }

    return false;
}
```

```cpp
auto renderables = m_EntityManager->GetAllEntitiesWithComponents<TransformComponent, SpriteComponent>();
for (const auto r : renderables)
{
    DrawEntity(r);
}
```