ENTITY

```
Entity(EntityType type = NONE);
Entity();
~Entity();
ecs::entity::EntityType getType() const;
void setId(size_t id);
size_t getId(void) const;
bool hasComponent(ecs::component::ComponentType type);
void addComponent(std::shared_ptr<ecs::component::IComponent> component);
std::shared_ptr<ecs::component::IComponent> getComponent(ecs::component::ComponentType type);
void removeComponent(ecs::component::ComponentType type);
```

ENTITYPOOL

```
EntityPool();
~EntityPool();
void AddEntity(ecs::entity::EntityType _EntityType);
void RemoveEntity(uint _Id);
std::vector<std::shared_ptr<ecs::entity::Entity>> &GetEntities();
std::string dumpDrawableEntities()
void Update();
```

IComponent

```
virtual ~IComponent() = default;
virtual ComponentType getComponentType() const = 0;
```

Speed

```
Speed(float speed = 1);
~Speed() override = default;
ComponentType getComponentType() const override;
float getSpeed() const;
void setSpeed(float speed);
```

Position

```
Position(float x = 0, float y = 0);
~Position() override = default;
ComponentType getComponentType() const override;
std::pair<float, float> getPosition() const;
void setX(float x);
void setY(float y);
```

Damage

```
Damage(int damage = 10);
~Damage() override = default;
ComponentType getComponentType() const override;
int getDamage() const;
void setDamage(int damage);
```

Collision

```
Collision(int width = 50, int height = 50);
~Collision() override = default;
ComponentType getComponentType() const override;
std::pair<int, int> getHitbox() const;
void setHitbox(std::pair<int, int> hitbox);
```

Health

```
Health(int health = 100);
~Health() override = default;
ComponentType getComponentType() const override;
int getHealth() const;
void setHealth(int health);
bool getIsAlive() const;
void setIsAlive(bool isAlive);
```

Drawable

```
Drawable(std::string _Texture, float _Scale);
~Drawable() override = default;
ComponentType getComponentType() const override;
float getScale() const;
void setScale(float _Scale);
std::string getTexture() const;
void setTexture(std::string _Texture);
```

Enum ComponentType

```
POSITIONCOMPONENT
HEALTHCOMPONENT
DAMAGECOMPONENT
SPEEDCOMPONENT
COLLISIONCOMPONENT
NONECOMPONENT
```

ISystem

```
virtual ~ISystem() = default;
virtual void update(std::vector<std::shared_ptr<ecs::entity::Entity>> &entities) = 0;
virtual SystemType getSystemType() const = 0;
```

PlayerSystem

```
PlayerSystem();
~PlayerSystem() override = default;
void update(std::vector<std::shared_ptr<ecs::entity::Entity>> &entities) override;
SystemType getSystemType() const override;
```

MovementSystem

```
Movement();
~Movement() override = default;
void update(std::vector<std::shared_ptr<ecs::entity::Entity>> &entities) override;
SystemType getSystemType() const override;
```

EnemySystem

```
EnemySystem();
~EnemySystem() override = default;
void update(std::vector<std::shared_ptr<ecs::entity::Entity>> &entities) override;
SystemType getSystemType() const override;
```

CollisionSystem

```
CollisionSystem();
~CollisionSystem() override = default;
void update(std::vector<std::shared_ptr<ecs::entity::Entity>> &entities) override;
SystemType getSystemType() const override;
```

DeathSystem

```
DeathSystem();
~DeathSystem() override = default;
void update(std::vector<std::shared_ptr<ecs::entity::Entity>> &entities) override;
SystemType getSystemType() const override;
```

EnemySystem

```
LaserSystem(ecs::component::Position pos, int owner, float speed);
~LaserSystem() override = default;
void update(std::vector<std::shared_ptr<ecs::entity::Entity>> &entities) override;
SystemType getSystemType() const override;
std::string getLaserString(void);
```

Enum SystemType

```
MOVEMENTSYSTEM
COLLISIONSYSTEM
PLAYERSYSTEM
ENEMYSYSTEM
NONESYSTEM
```