Rendering Football Data in 3D

with Rust and Bevy

Joris Bekkers @UnravelSports unravelsports.github.io

UNRAVEL SPORTS

Contents

- Introduction
- Why 3D?
- Why Rust?
- Project

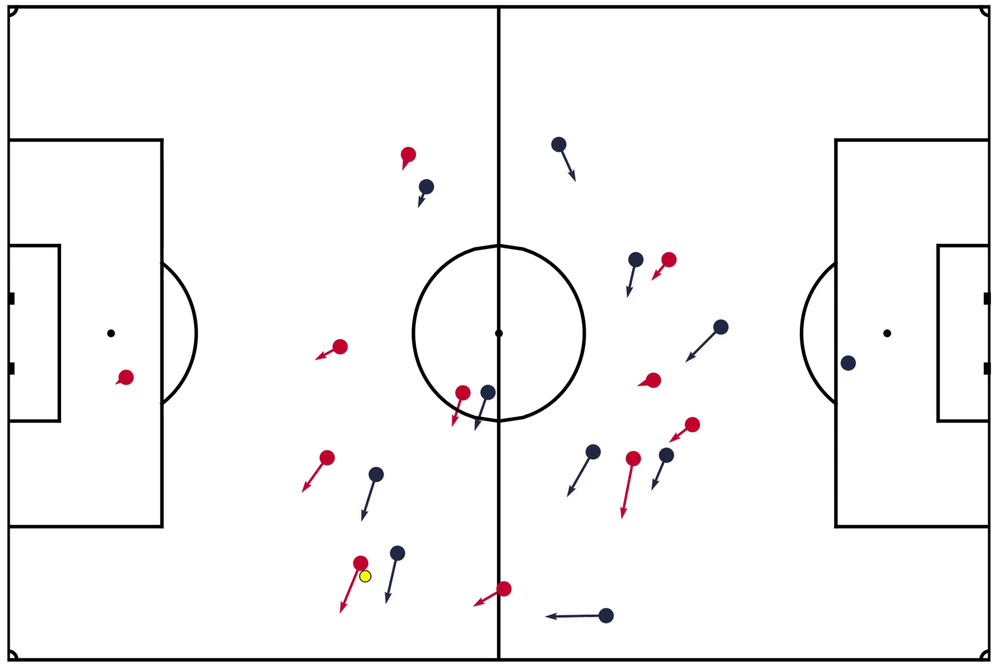
Introduction

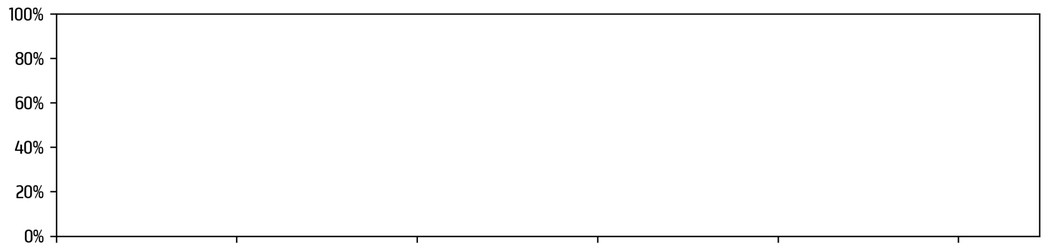
- Joris Bekkers
- Football Analytics Consultant
- Python, SQL etc.
- @UnravelSports
- unravelsports.github.io

Why 3D?

- Body Pose / Skeletal Data
- NBA Hawk-eye Deal

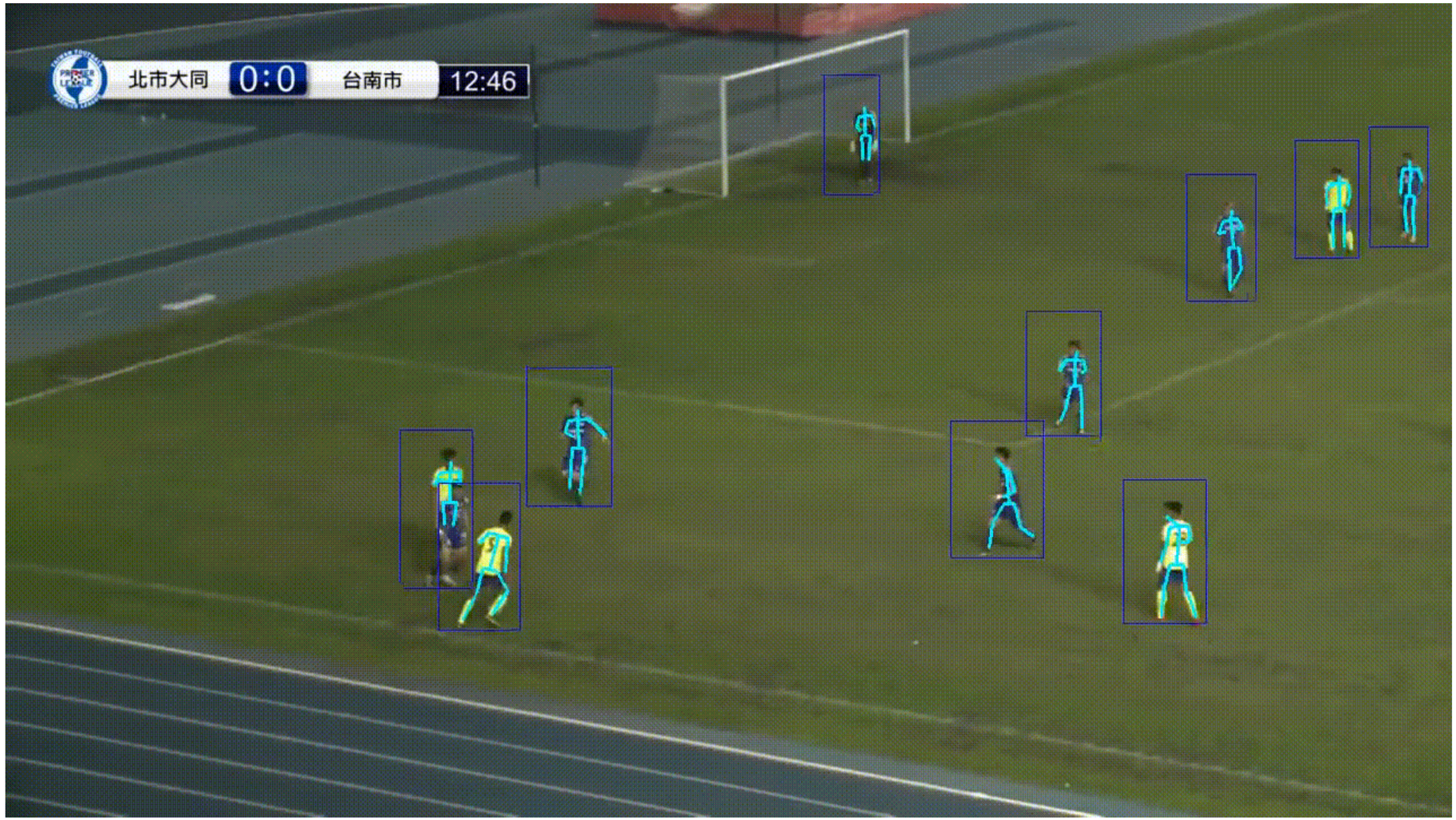
  https://pr.nba.com/nba-sony-hawk-eye-innovations-partnership/
- Any camera angle (Player POV)
- Virtual Reality





Sahasrabudhe & Bekkers (2023)

Why 3D?

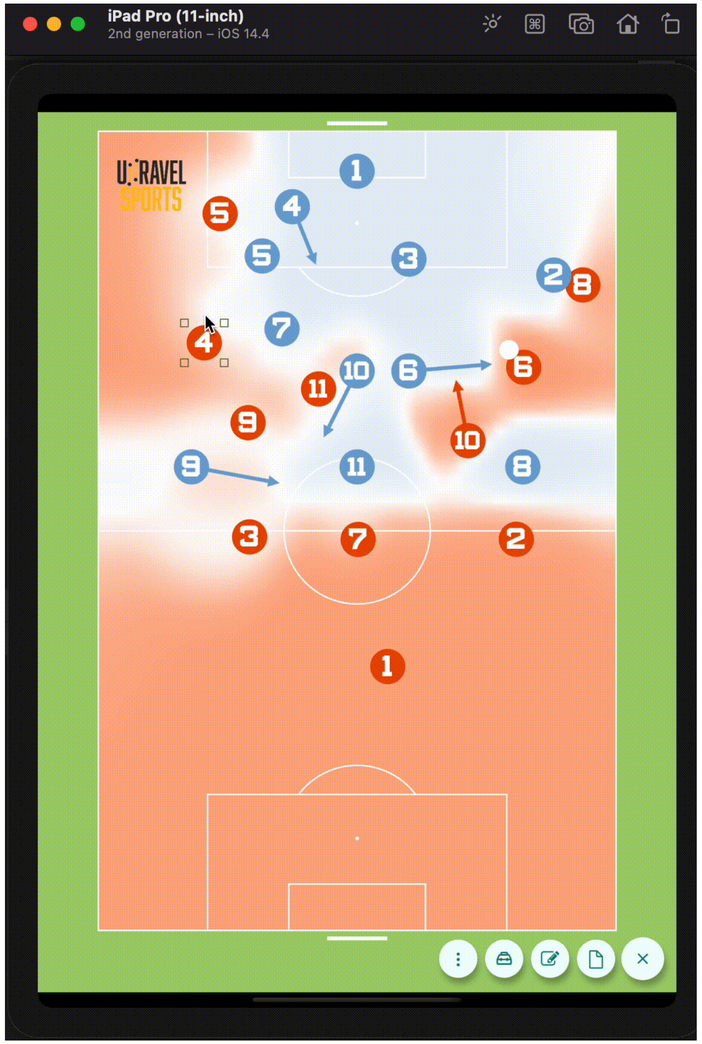


Andrea Pennisi (https://www.youtube.com/watch?v=Z-39qwJWQ6M)

Why Rust?

- Modern Language (2015)
- A lot faster than Python
- Simplified syntax compared to C / C++
- Always good to learn a new language

  Flutter / Dart Mobile App (2021)



UNRAVEL SPORTS

Preview

3D Animated Football Data

Joris Bekkers @UnravelSports unravelsports.github.io

UNRAVEL SPORTS

Preview

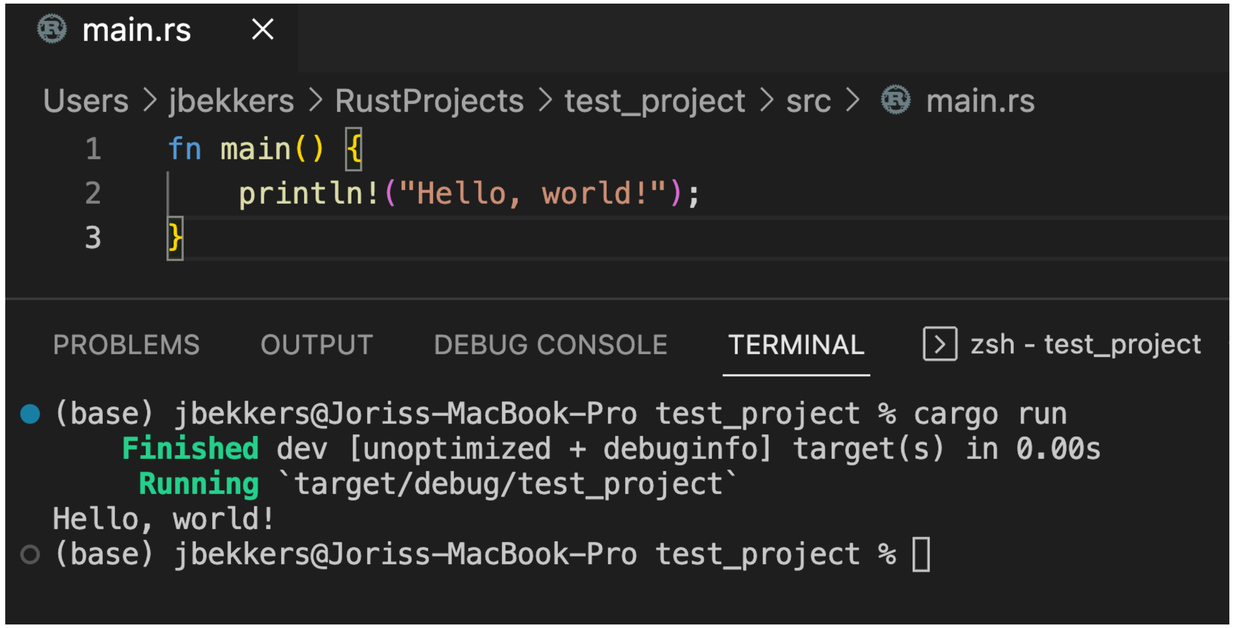
Rust

The Very Basics

Joris Bekkers @UnravelSports unravelsports.github.io

UNRAVEL SPORTS

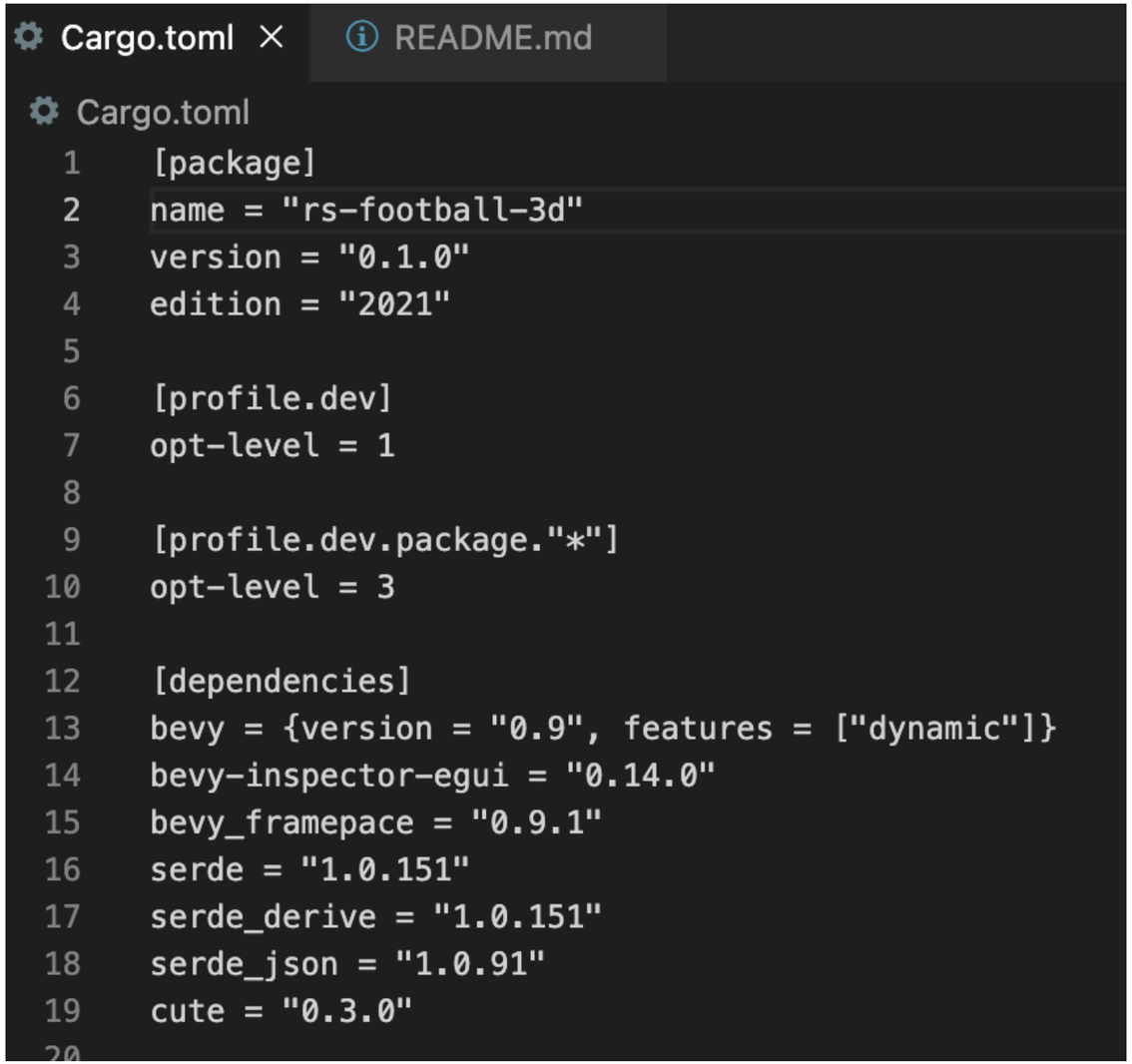
Rust Basics



- Start a new Rust project ➤ `cargo new [your_project_name]`
- Build your project ➤ `cargo build`
- Run your project ➤ `cargo run`

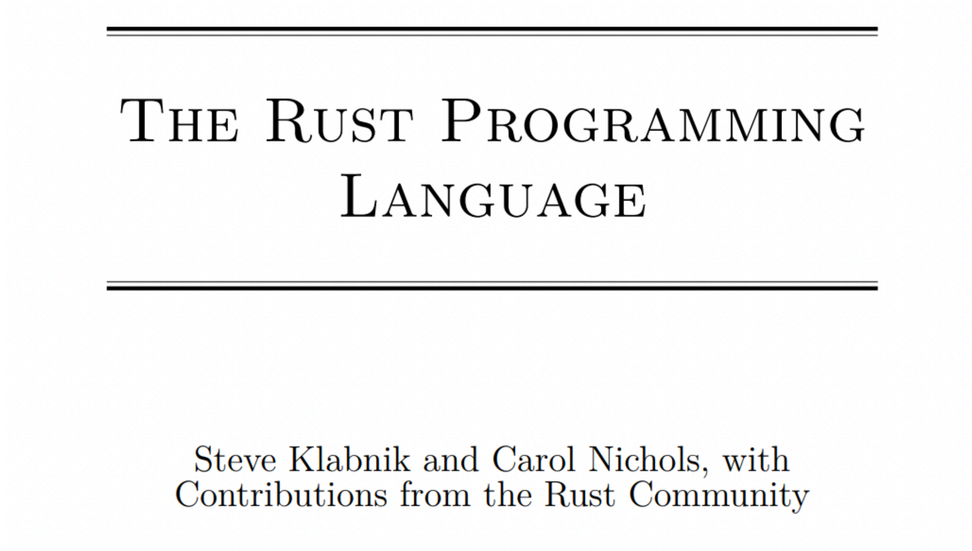
UNRAVEL SPORTS

Rust Basics



UNRAVEL SPORTS

Rust



- Some major differences Rust v Python:
  - Statically typed
  - Memory Management
  - Syntax

Bevy

Free & Open-source Game Engine

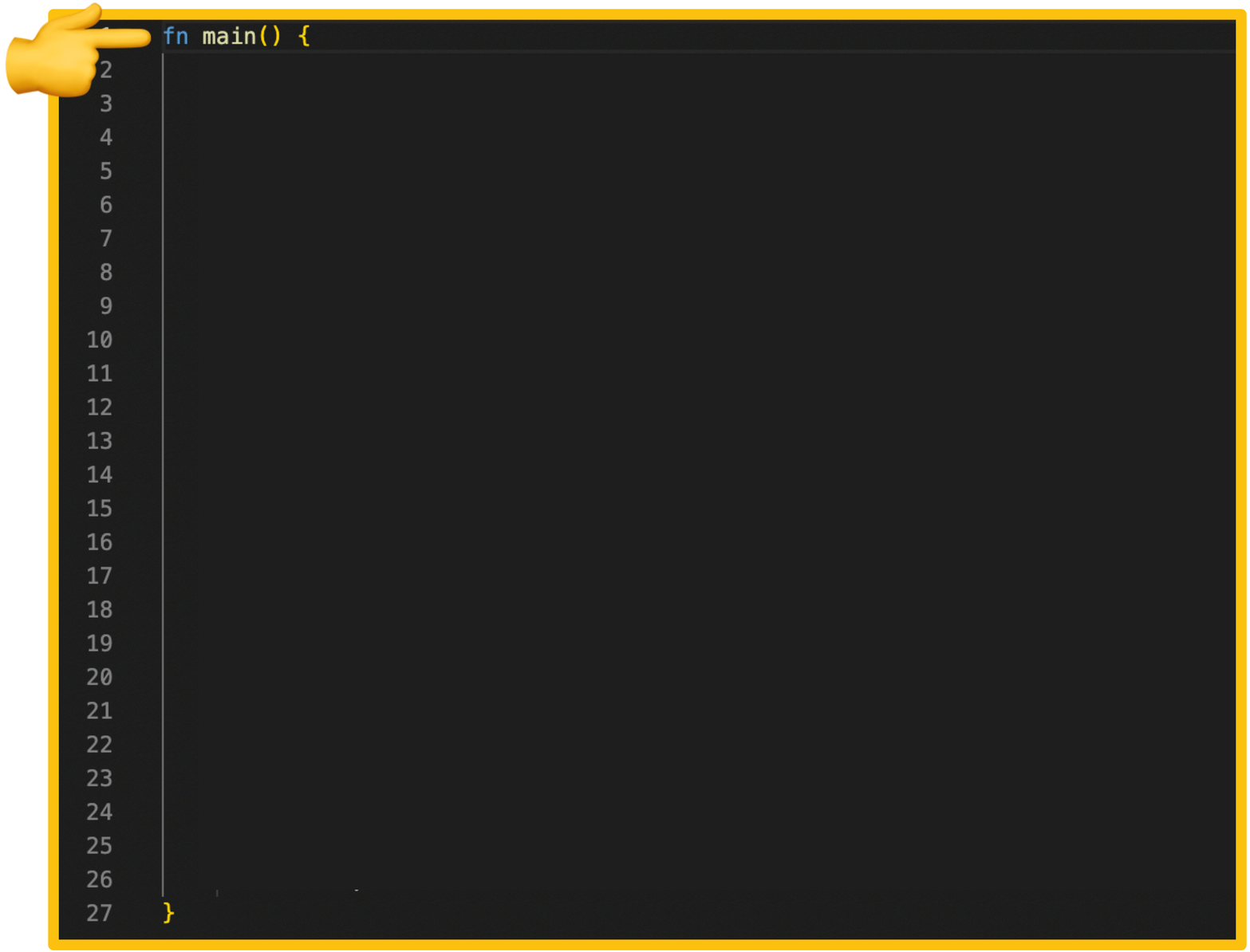
Joris Bekkers @UnravelSports unravelsports.github.io

Bevy Basics

- Only a couple years old
- 2D & 3D Games
- Cross Platform
  - Windows, MacOS, Linux, Web (Android & iOS on the way)

UNRAVEL SPORTS

Bevy Basics

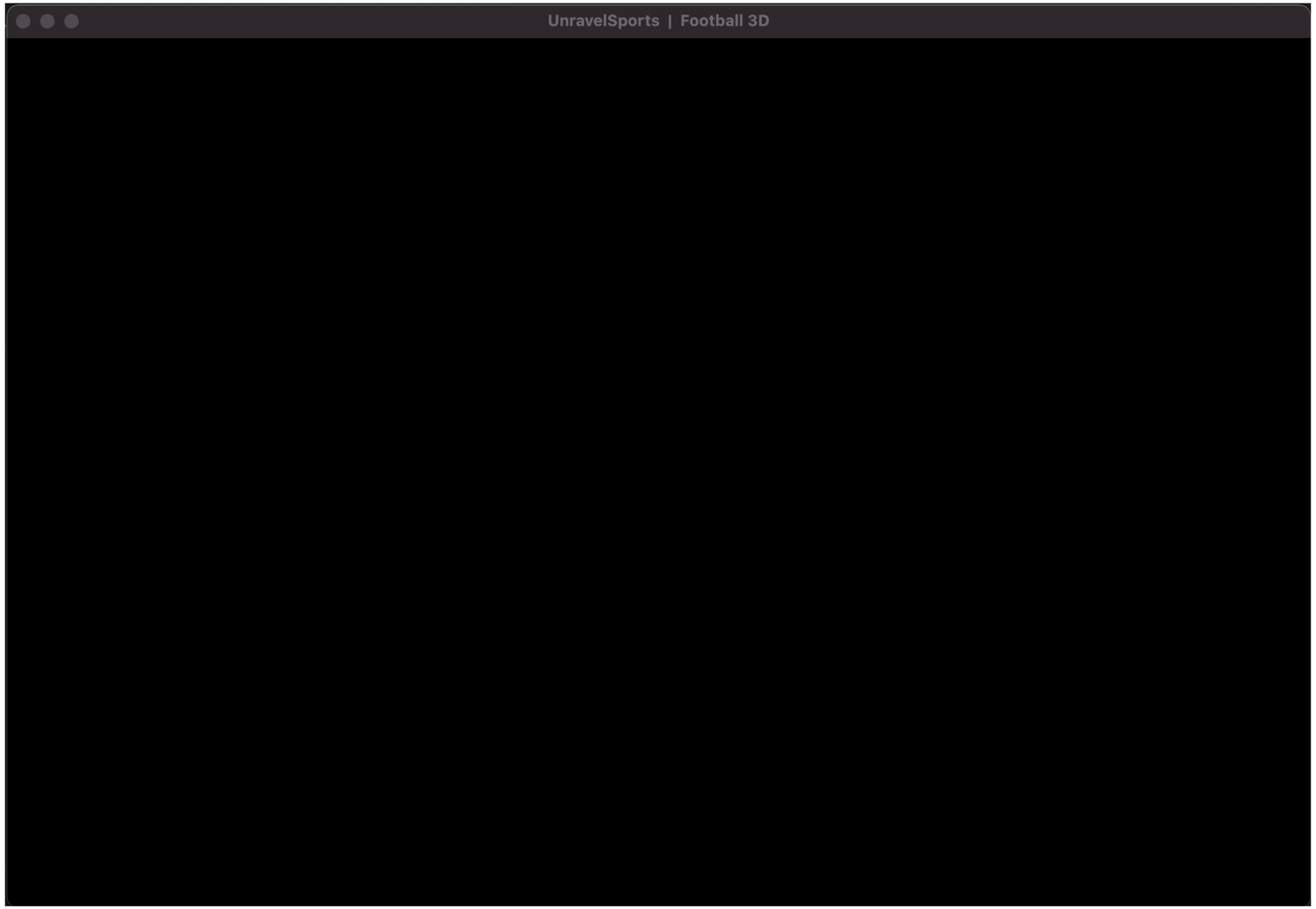


UNRAVEL SPORTS

Bevy Basics

UNRAVEL SPORTS

Bevy Basics



UNRAVEL SPORTS

Bevy Basics

```rust
fn main() {
    App::new()
        // create window
        .add_plugins(DefaultPlugins.set(WindowPlugin {
            window: WindowDescriptor {
                width: WIDTH,
                height: HEIGHT,
                title: "UnravelSports  |  Football 3D".to_string(),
                resizable: false,
                ..Default::default()
            },
            ..default()
        }))
        // systems before start up:
        .add_startup_system_to_stage(StartupStage::PreStartup, asset_loading)
        // systems at start up:
        .add_startup_system(build_basic_scene)
        // systems to run each frame
        .add_system(update)
        // run app
        .run();
}
```

UNRAVEL SPORTS

Bevy Basics

```rust
fn main() {
    App::new()
        // create window
        .add_plugins(DefaultPlugins.set(WindowPlugin {
            window: WindowDescriptor {
                width: WIDTH,
                height: HEIGHT,
                title: "UnravelSports  |  Football 3D".to_string(),
                resizable: false,
                ..Default::default()
            },
            ..default()
        }))



        // systems before start up:
        .add_startup_system_to_stage(StartupStage::PreStartup, asset_loading)
        // systems at start up:
        .add_startup_system(build_basic_scene)
        // systems to run each frame
        .add_system(update)
        // run app
        .run();
}
```

UNRAVEL SPORTS

Bevy Basics

```rust
fn main() {
    App::new()
        // create window
        .add_plugins(DefaultPlugins.set(WindowPlugin {
            window: WindowDescriptor {
                width: WIDTH,
                height: HEIGHT,
                title: "UnravelSports  |  Football 3D".to_string(),
                resizable: false,
                ..Default::default()
            },
            ..default()
        }))



        // systems before start up:
        .add_startup_system_to_stage(StartupStage::PreStartup, asset_loading)
        // systems at start up:
        .add_startup_system(build_basic_scene)
        // systems to run each frame
        .add_system(update)
        // run app
        .run();
}
```

Bevy Basics

UNRAVEL SPORTS

Bevy Basics

```rust
fn main() {
    App::new()
        // create window
        .add_plugins(DefaultPlugins.set(WindowPlugin {
            window: WindowDescriptor {
                width: WIDTH,
                height: HEIGHT,
                title: "UnravelSports  |  Football 3D".to_string(),
                resizable: false,
                ..Default::default()
            },
            ..default()
        }))
        // insert a light source
        .insert_resource(AmbientLight {
            color: Color::WHITE,
            brightness: 1.0,
        })
        // systems before start up:
        .add_startup_system_to_stage(StartupStage::PreStartup, asset_loading)
        // systems at start up:
        .add_startup_system(build_basic_scene)
        // systems to run each frame
        .add_system(update)
        // run app
        .run();
}
```

Bevy Basics

UNRAVEL SPORTS

Project Assets

3D Models & Tracking Data

UNRAVEL SPORTS

Project Assets

UNRAVEL SPORTS

Project Assets

- SketchFab.com / TurboSquid.com / CGTrader.com
- Use Blender (open-source 3D modeling software) to:
  - Change model (scale, looks etc.)
  - Convert from .blend to .gltf / .glb

UNRAVEL SPORTS

Project Assets

UNRAVEL SPORTS

3D Model from NGUYENNGHIAKK (www.sketchfab.com)

Loading Assets

3D Models & Tracking Data

Joris Bekkers @UnravelSports unravelsports.github.io

UNRAVEL SPORTS

Loading Assets

```rust
fn main() {
    App::new()
        // create window
        .add_plugins(DefaultPlugins.set(WindowPlugin {
            window: WindowDescriptor {
                width: WIDTH,
                height: HEIGHT,
                title: "UnravelSports  |  Football 3D".to_string(),
                resizable: false,
                ..Default::default()
            },
            ..default()
        }))
        // insert a light source
        .insert_resource(AmbientLight {
            color: Color::WHITE,
            brightness: 1.0,
        })
        // systems before start up:
        .add_startup_system_to_stage(StartupStage::PreStartup, asset_loading)
        // systems at start up:
        .add_startup_system(build_basic_scene)
        // systems to run each frame
        .add_system(update)
        // run app
        .run();
}
```

UNRAVEL SPORTS

Loading Assets

```
fn asset_loading(mut commands: Commands, assets: Res<AssetServer>) {
    commands.insert_resource(GameAssets {
        goal_scene: assets.load("goal_frame.glb#Scene0"),
        home_player_scene: assets.load("player_blue.glb#Scene0"),
        away_player_scene: assets.load("player_red.glb#Scene0"),
        stadium_scene: assets.load("stadium.glb#Scene0"),
        pitch_scene: assets.load("pitch.glb#Scene0"),
        play_button: assets.load("play_pause.png"),
    });
}
```

- Commands used to:
  - Add/remove components
  - Move components
  - Modify components

UNRAVEL SPORTS

Loading Assets

```
fn asset_loading(mut commands: Commands, assets: Res<AssetServer>) {
    commands.insert_resource(GameAssets {
        goal_scene: assets.load("goal_frame.glb#Scene0"),
        home_player_scene: assets.load("player_blue.glb#Scene0"),
        away_player_scene: assets.load("player_red.glb#Scene0"),
        stadium_scene: assets.load("stadium.glb#Scene0"),
        pitch_scene: assets.load("pitch.glb#Scene0"),
        play_button: assets.load("play_pause.png"),
    });
}
```

- AssetServer used to load:
  - Images
  - 3D Models
  - Sounds

UNRAVEL SPORTS

Loading Assets

```
72  fn asset_loading(mut commands: Commands, assets: Res<AssetServer>) {
73      commands.insert_resource(GameAssets {
74          goal_scene: assets.load("goal_frame.glb#Scene0"),
75          home_player_scene: assets.load("player_blue.glb#Scene0"),
76          away_player_scene: assets.load("player_red.glb#Scene0"),
77          stadium_scene: assets.load("stadium.glb#Scene0"),
78          pitch_scene: assets.load("pitch.glb#Scene0"),
79          play_button: assets.load("play_pause.png"),
80      });
81  }
```

- Insert Components

UNRAVEL SPORTS

Loading Data

```
fn main() {
    // create window
    App::new()
        // plugins:
        .add_plugins(DefaultPlugins.set(WindowPlugin {
            window: WindowDescriptor {
                width: WIDTH,
                height: HEIGHT,
                title: "UnravelSports  |  Football 3D".to_string(),
                resizable: false,
                ..Default::default()
            },
            ..default()
        }))
        // insert light source
        .insert_resource(AmbientLight {
            color: Color::WHITE,
            brightness: 1.0,
        })
        // systems before start up:
        .add_startup_system_to_stage(StartupStage::PreStartup, asset_loading)
        .add_startup_system_to_stage(StartupStage::PreStartup, json_loading)
        // systems at start up:
        .add_startup_system(build_basic_scene)
        // systems to run each frame
        .add_system(update)
        // run app
        .run();
}
```

UNRAVEL SPORTS

Loading Data

```
{
    "x": -44.02,
    "y": 3.73,
    "vx": -0.886,
    "vy": 0.343,
    "pid": 1,
    "team": "home"
},
```

```
10  pub struct PlayerFrame {
11      pub x: f32,
12      pub y: f32,
13      pub vx: f32,
14      pub vy: f32,
15      pub pid: u32,
16      pub team: String
17  }
```

- Coordinate data for every Player and Ball
  - 10 Frames per Second
  - Center of pitch at (0, 0)

Building A Basic Scene

With 3D Models

Joris Bekkers @UnravelSports unravelsports.github.io

UNRAVEL SPORTS

Project Assets

```rust
fn main() {
    // create window
    App::new()
        // plugins:
        .add_plugins(DefaultPlugins.set(WindowPlugin {
            window: WindowDescriptor {
                width: WIDTH,
                height: HEIGHT,
                title: "UnravelSports  |  Football 3D".to_string(),
                resizable: false,
                ..Default::default()
            },
            ..default()
        }))
        // insert light source
        .insert_resource(AmbientLight {
            color: Color::WHITE,
            brightness: 1.0,
        })
        // systems before start up:
        .add_startup_system_to_stage(StartupStage::PreStartup, asset_loading)
        .add_startup_system_to_stage(StartupStage::PreStartup, json_loading)
        // systems at start up:
        .add_startup_system(build_basic_scene)
        // systems to run each frame
        .add_system(update)
        // run app
        .run();
}
```

UNRAVEL SPORTS

Building Basic Scene

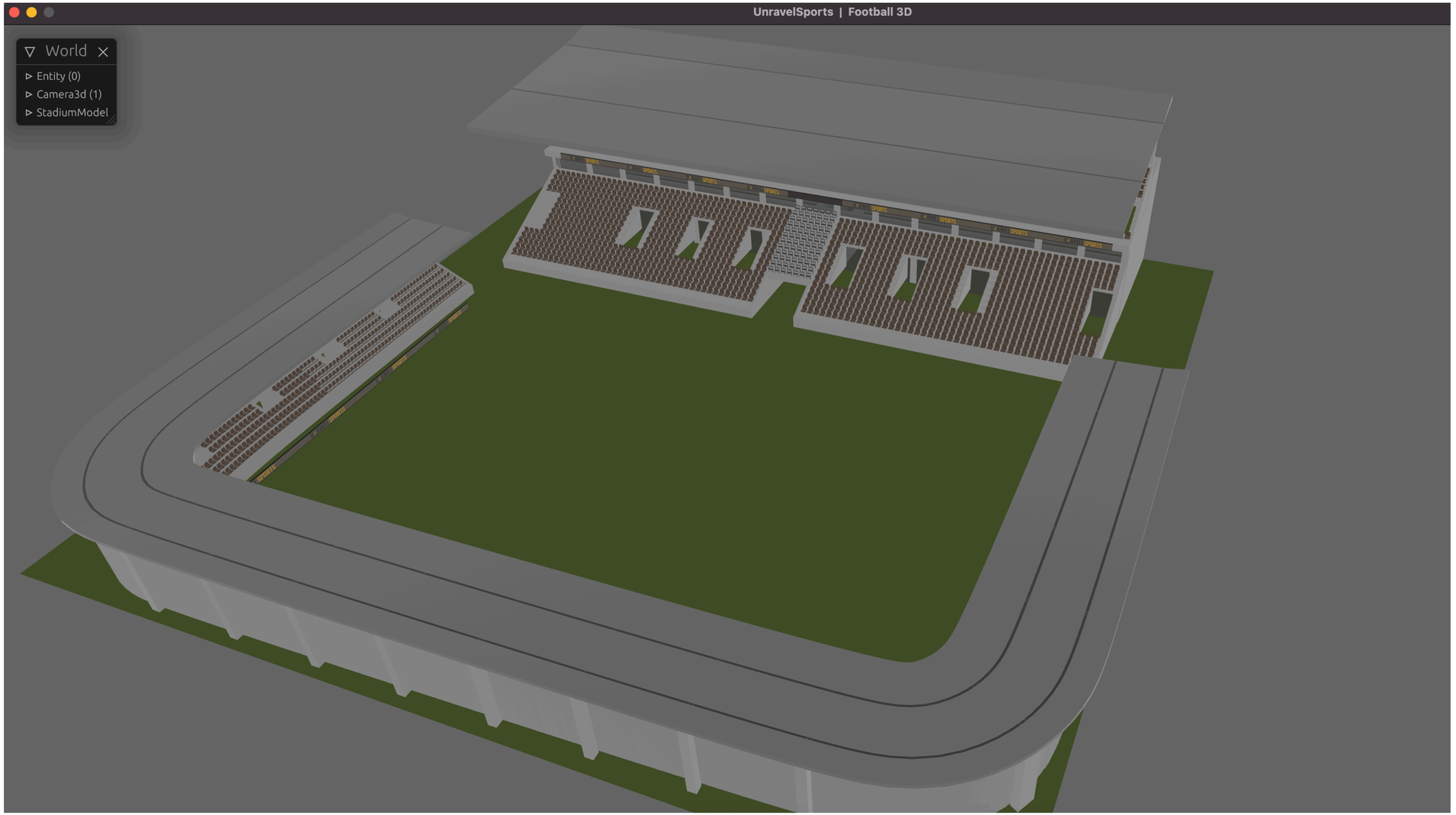
```rust
fn build_basic_scene(
    mut commands: Commands,
    mut meshes: ResMut<Assets<Mesh>>,
    mut materials: ResMut<Assets<StandardMaterial>>,
    game_assets: Res<GameAssets>,
    match_data: ResMut<MatchData>
) {

    commands
        .spawn(SceneBundle {
            scene: game_assets.stadium_scene.clone(),
            transform: Transform::from_xyz(-8.0, -0.5, -7.7),
            ..Default::default()
        })
        .insert(Name::new("StadiumModel"));

```

- Basic Scene is like a game level or map

Building Basic Scene



Building Basic Scene

Building A Basic Scene

Adding Player Models

UNRAVEL SPORTS

Building Basic Scene

```
    let idx: usize = 0;
    for p in &match_data.data.players[idx]{
        println!("{:?}", p);
157
158
```

UNRAVEL SPORTS

Building Basic Scene

```
155     let idx: usize = 0;
156     for p in &match_data.data.players[idx]{
157         println!("{:?}", p);
158
159         let scene = if p.team == "home" {
160             game_assets.home_player_scene.clone()
161         } else {
162             game_assets.away_player_scene.clone()
163         };
164
```

UNRAVEL SPORTS

Building Basic Scene

```rust
    let idx: usize = 0;
    for p in &match_data.data.players[idx]{
        println!("{:?}", p);

        let scene = if p.team == "home" {
            game_assets.home_player_scene.clone()
        } else {
            game_assets.away_player_scene.clone()
        };

```

```
{
    "x": -44.02,
    "y": 3.73,
    "vx": -0.886,
    "vy": 0.343,
    "pid": 1,
    "team": "home"
},
```

UNRAVEL SPORTS

Building Basic Scene

```
155     let idx: usize = 0;
156     for p in &match_data.data.players[idx]{
157         println!("{:?}", p);
158
159         let scene = if p.team == "home" {
160             game_assets.home_player_scene.clone()
161         } else {
162             game_assets.away_player_scene.clone()
163         };
164
```

```
{
    "x": -44.02,
    "y": 3.73,
    "vx": -0.886,
    "vy": 0.343,
    "pid": 1,
    "team": "home"
},
```

```
72  fn asset_loading(mut commands: Commands, assets: Res<AssetServer>) {
73      commands.insert_resource(GameAssets {
74          goal_scene: assets.load("goal_frame.glb#Scene0"),
75          home_player_scene: assets.load("player_blue.glb#Scene0"),
76          away_player_scene: assets.load("player_red.glb#Scene0"),
77          stadium_scene: assets.load("stadium.glb#Scene0"),
78          pitch_scene: assets.load("pitch.glb#Scene0"),
79          play_button: assets.load("play_pause.png"),
80      });
81  }
```

UNRAVEL SPORTS

Building Basic Scene

```rust
    let idx: usize = 0;
    for p in &match_data.data.players[idx]{
        println!("{:?}", p);

        let scene = if p.team == "home" {
            game_assets.home_player_scene.clone()
        } else {
            game_assets.away_player_scene.clone()
        };

        let theta = p.vy / p.vx;
        commands
            .spawn(SceneBundle {
                scene: scene,
                transform: Transform::from_xyz(p.x, 0.0, -1.0*p.y)
                .with_rotation(Quat::from_rotation_y(theta.tan())),

                ..Default::default()
            })
            .insert(Player {
                pid: p.pid
            })
            .insert(Name::new(format!("Player-{t}-{p}", t=p.team, p=p.pid)));

    }
```


```
{
    "x": -44.02,
    "y": 3.73,
    "vx": -0.886,
    "vy": 0.343,
    "pid": 1,
    "team": "home"
},
```

UNRAVEL SPORTS

Building Basic Scene

```rust
    let idx: usize = 0;
    for p in &match_data.data.players[idx]{
        println!("{:?}", p);

        let scene = if p.team == "home" {
            game_assets.home_player_scene.clone()
        } else {
            game_assets.away_player_scene.clone()
        };

        let theta = p.vy / p.vx;
        commands
            .spawn(SceneBundle {
                scene: scene,
                transform: Transform::from_xyz(p.x, 0.0, -1.0*p.y)
                .with_rotation(Quat::from_rotation_y(theta.tan())),

                ..Default::default()
            })
            .insert(Player {
                pid: p.pid
            })
            .insert(Name::new(format!("Player-{t}-{p}", t=p.team, p=p.pid)));

    }
```


```
{
    "x": -44.02,
    "y": 3.73,
    "vx": -0.886,
    "vy": 0.343,
    "pid": 1,
    "team": "home"
},
```

UNRAVEL SPORTS

Building Basic Scene

```
155     let idx: usize = 0;
156     for p in &match_data.data.players[idx]{
157         println!("{:?}", p);
158
159         let scene = if p.team == "home" {
160             game_assets.home_player_scene.clone()
161         } else {
162             game_assets.away_player_scene.clone()
163         };
164
165         let theta = p.vy / p.vx;
166         commands
167             .spawn(SceneBundle {
168                 scene: scene,
169                 transform: Transform::from_xyz(p.x, 0.0, -1.0*p.y)
170                 .with_rotation(Quat::from_rotation_y(theta.tan())),
171
172                 ..Default::default()
173             })
174             .insert(Player {
175                 pid: p.pid
176             })
177             .insert(Name::new(format!("Player-{t}-{p}", t=p.team, p=p.pid)));
178
179     }
```

```
pub struct Player {
    pub pid: u32
}
```

UNRAVEL SPORTS

Building Basic Scene

```
155     let idx: usize = 0;
156     for p in &match_data.data.players[idx]{
157         println!("{:?}", p);
158 
159         let scene = if p.team == "home" {
160             game_assets.home_player_scene.clone()
161         } else {
162             game_assets.away_player_scene.clone()
163         };
164 
165         let theta = p.vy / p.vx;
166         commands
167             .spawn(SceneBundle {
168                 scene: scene,
169                 transform: Transform::from_xyz(p.x, 0.0, -1.0*p.y)
170                 .with_rotation(Quat::from_rotation_y(theta.tan())),
171 
172                 ..Default::default()
173             })
174             .insert(Player {
175                 pid: p.pid
176             })
177             .insert(Name::new(format!("Player-{t}-{p}", t=p.team, p=p.pid)));
178 
179     }
```

```
{
    "x": -44.02,
    "y": 3.73,
    "vx": -0.886,
    "vy": 0.343,
    "pid": 1,
    "team": "home"
},
```

UNRAVEL SPORTS

Building Basic Scene

Updating the Scene

Changing Player & Ball Positions

Joris Bekkers @UnravelSports unravelsports.github.io

UNRAVEL SPORTS

Updating the Scene

```
fn main() {
    // create window
    App::new()
        // plugins:
        .add_plugins(DefaultPlugins.set(WindowPlugin {
            window: WindowDescriptor {
                width: WIDTH,
                height: HEIGHT,
                title: "UnravelSports  |  Football 3D".to_string(),
                resizable: false,
                ..Default::default()
            },
            ..default()
        }))
        // insert light source
        .insert_resource(AmbientLight {
            color: Color::WHITE,
            brightness: 1.0,
        })
        // systems before start up:
        .add_startup_system_to_stage(StartupStage::PreStartup, asset_loading)
        .add_startup_system_to_stage(StartupStage::PreStartup, json_loading)
        // systems at start up:
        .add_startup_system(build_basic_scene)
        // systems to run each frame
        .add_system(update)
        // run app
        .run();
}
```

UNRAVEL SPORTS

Update the Scene

```
192  fn update(
     mut players: Query<(&Player, &mut Transform)>,
     buttons: Query<&Button>,
195  mut match_data: ResMut<MatchData>,
196  time: Res<Time>,
197  ) {
```

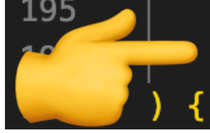
UNRAVEL SPORTS

Update the Scene

```
192  fn update(
193      mut players: Query<(&Player, &mut Transform)>,
194      buttons: Query<&Button>,
     
     mut match_data: ResMut<MatchData>,
     time: Res<Time>,
197  ) {
```

Update the Scene

```
192    fn update(
193        mut players: Query<(&Player, &mut Transform)>,
194        buttons: Query<&Button>,
195        mut match_data: ResMut<MatchData>,
           time: Res<Time>,
       ) {
```

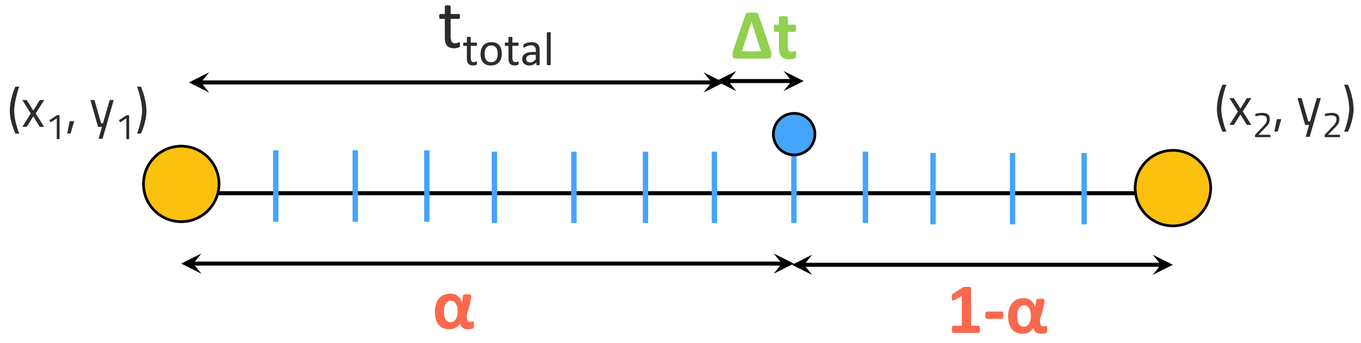


- Football tracking data is 10FPS
- Game-engine updates between 30 and 60 FPS

Update the Scene

```
192  fn update(
193      mut players: Query<(&Player, &mut Transform)>,
194      buttons: Query<&Button>,
195      mut match_data: ResMut<MatchData>,
196      time: Res<Time>,
197  ) {
         let (idx, alpha) = match_data.get_interpolation_values_and_increment(time.delta_seconds());
```

- Interpolate between (x_1, y_1) and next frame (x_2, y_2) **10FPS** data

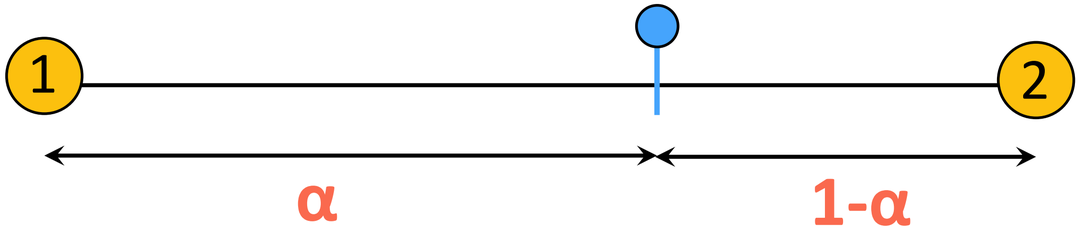


UNRAVEL SPORTS

Update the Scene

```
192  fn update(
193      mut players: Query<(&Player, &mut Transform)>,
194      buttons: Query<&Button>,
195      mut match_data: ResMut<MatchData>,
196      time: Res<Time>,
197  ) {
         let (idx, alpha) = match_data.get_interpolation_values_and_increment(time.delta_seconds());
```

- Interpolate between (x_1, y_1) and next frame (x_2, y_2) **10FPS** data
- Exact coordinate at an **in-between frame**
  - **α** (x_1, y_1) + (**1 - α**) (x_2, y_2)



UNRAVEL SPORTS

Update the Scene

```
// interpolate x and z by taking the value from the current frame and the next frame;
transform.translation.x = interpolate(alpha, p.x, p1.x);
transform.translation.z = interpolate(alpha, p.y, p1.y) * -1.0;

let vx: f32 = interpolate(alpha, p.vx, p1.vx);
let vy: f32 = interpolate(alpha, p.vy, p1.vy);
let theta = vy / vx;

// only update rotation if player speed is more than 5ms, because lower results in some buggy/spinny players
if (vx.powi(2) + vy.powi(2)).sqrt() > 5.0 {
    let rot = transform.rotation.y;
    transform.rotate_y(theta.tan() - rot);
}
```

UNRAVEL SPORTS

Update the Scene

```
// interpolate x and z by taking the value from the current frame and the next frame;
transform.translation.x = interpolate(alpha, p.x, p1.x);
transform.translation.z = interpolate(alpha, p.y, p1.y) * -1.0;

let vx: f32 = interpolate(alpha, p.vx, p1.vx);
let vy: f32 = interpolate(alpha, p.vy, p1.vy);
let theta = vy / vx;

// only update rotation if player speed is more than 5ms, because lower results in some buggy/spinny players
if (vx.powi(2) + vy.powi(2)).sqrt() > 5.0 {
    let rot = transform.rotation.y;
    transform.rotate_y(theta.tan() - rot);
}
```

UNRAVEL SPORTS

Result

Animated Football Data in 3D

UNRAVEL SPORTS

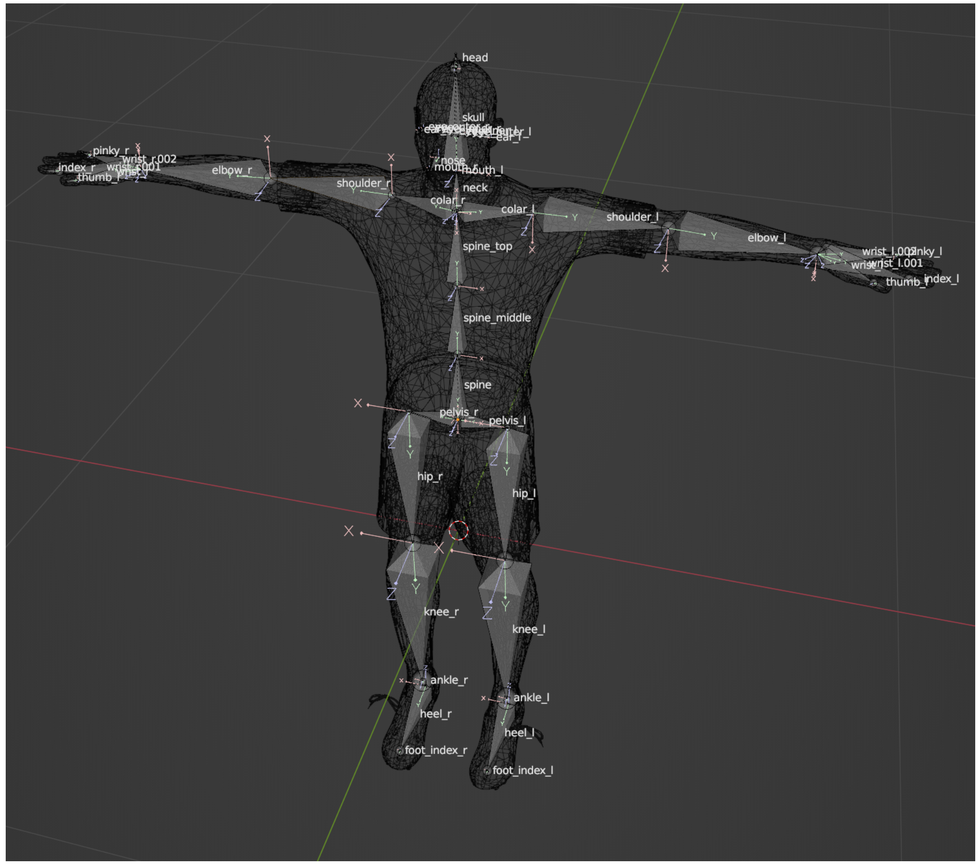
Result

Player Point of View

UNRAVEL SPORTS

Future Work

- Add 3D body pose animation
- Load game moments from API
- Improve interface
- Improve camera controls
- Virtual Reality



GitHub

- github.com/UnravelSports/rs-football-3d

References & Resources

- Rust Language Open-Source Book

  https://github.com/rust-lang/book

- Basic Bevy Development by LogicProjects

  https://www.youtube.com/@logicprojects

- Bevy Tower Defense Game by LogicProjects

  https://github.com/mwbryant/bevy-tower-defense-tutorial

The End!

Questions?