
File Edit Window Tools Build Select Actor Help
GOALS
Selection Mode
Platforms
Perspective Lit Show Scalability: High
Previewing ABP_GoalsCharacter_C
LOD: 0
Current Screen Size 1.289
JUST FOOTBALL
GOALS
FOR THE PLAYER
GAMEPLAY FIRST
FREE TO PLAY
CROSS PLATFORM
FTP

Seed-based Character Generation in UE5

Mario Caprino - GOALS

GOALS

- New free to play soccer game
- Focus on responsive and fair gameplay
- E-sports ready
- To be released 2026

Generated characters

- No licensed players or clubs
- All characters are generated on demand at runtime from seed
- Character visuals must remain deterministic

EDIT TEAM

SQUAD | COLLECTION

mario+test1

3,264 0

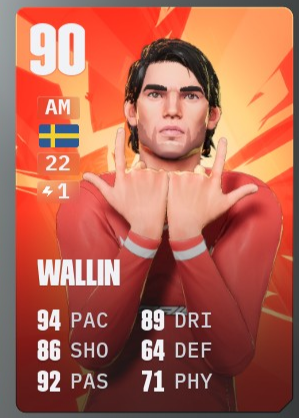
FILTER | ACTIVE PLAYERS

MAJEWSKI

86 OVR

93 PAC
89 SHO
83 PAS
74 DRI
37 DEF
82 PHY

176 CM
79 KG
PF



Unreal’s Mutable plugin

- Seed based character generation is a main feature of GOALS
- Mutable is a beta feature
- Makes us dependent on an external providers road map
- Better flexibility with our own solution

Toolchain overview

1) Populate string map from character seed
2) Convert strings to visual data using blueprints
3) Combine visual data to Unreal skeletal mesh

Dividing character generation into multiple steps allow us to provide artist tools for each level

Rendering Preset
Load Presset
SavePresset
Card Rendering Preset
DA_CardRenderPreset_Gener
Pose
AS_CardPose_Generic5_v1
Face Pose
AS_CardPose_Generic5_Face
Card Frame Breakout
Level Sequence
Generate Level Sequence
Link Animations
Open Level Sequence
Level Sequence
LS_CardPose_Generic5
Lighting Preset
SaveLightingPreset
Spawn Spot Light
Lighting Presset
DA_CardLighting_Generic5
Selected Light
Use Tier Color as Light...

Preview Options
Randomize
Preview Tier: Basic
Seed: 1731776128
Is Female
Gen Version: 1
Bypass Face Joint Scaling
Nationality: namibia
Preview Kit
Debug Options

character
takashi
eye_color
eye_color_hazel
hair_color
hair_color_brown_darkintense
skin_color
sc_af_05
hair_style
Undercut_01
facial_hair
moustache_01
eyebrows
eyebrows_01_m
shoes
shoes_01a_v01
eyelashes
eyelashes01_v01_m
gloves
none
nose_profile
nubian
sleeves
none
jersey
light01_v03
shorts
light01_v03
socks
light01_v03
kit_style
default
baked_expression


Help
GOALS Tools
Select
Actor
Help
GOALS Tools
Platforms
Platforms
EUW Play...r Editor
Gear Item Preset
DA_Jersey_England01_Longsleeve
Preview General
Preview LOD
Preview Player Marking
Preview Player Marking
Player Name
LONG. NAME
Player Number
86
Player Name Font
SF_Inter24ptSemiBold
Player Number Font
SF_GoalsEnglish
General
Material
Decals
Decals
Decal Socket Set
DA_Decals_Jersey_02
Player Number 0
Sponsor 0
Team Crest
Player Name
Default
Decal Params
Active
Decal Type
Single Channel Sdf
Default Decal Map
T_World_S21_SDF
Decal Style
Color 0
Color 1
Color 2
Color 3
Roughness
0.3064
Metallic
0.0
Normal Opacity
1.0
Bump Height
0.1
Bump Width
0.1
Shadow Opacity
0.5
Shadow Width
0.5
Scale
1.0
Offset
0.0
0.0
Rotation
0.0
Bend Amount
0.0
Flip Symmetry

Nationalities

- 186 countries
- France: Mediterranean 85%, Arab 10%, Bantoid 3%
- England: Germanic 81%, Asian 9%, Bantoid 4%, Mulatto 3%
- Spain: Mediterranean 94%, Bantoid 3%, Arab 2%, Slavic 1%
- Germany: Germanic 92%, Turkic 4%, Slavic 2%, Bantoid 2%

Ethnicity groups

- 15 ethnicity groups
- Amerindian, Arab, Baltic, Bantoid, Brazilian, EastAsian, Germanic, Indian, Iranian, Mediterranean, Nordic, Sahelian, Slavic, Turkic, Uralic
- Ethnicity groups describe common visual features
- Base head, Eye color, Hair color, Skin color, Hair style, Facial hair, Eyebrows, Eyelashes, Nose profile

Base Head

Add new item

caucasian_male_01 L:33% B:33% S:33% (wt:1.0) Everyone

caucasian_male_03 L:33% B:33% S:33% (wt:1.0) Everyone

caucasian_male_02 L:33% B:33% S:33% (wt:1.0) Everyone

caucasian_male_05 L:0% B:0% S:0% (wt:0.0) Everyone

Add new item

caucasian_female_01 L:33% B:33% S:33% (wt:1.0) Everyone

caucasian_female_03 L:33% B:33% S:33% (wt:1.0) Everyone

caucasian_female_04 L:33% B:33% S:33% (wt:1.0) Everyone

Eye Color

eye_color_light_blue L:17% B:17% S:17% (wt:3.0) Everyone

eye_color_blue L:33% B:33% S:33% (wt:6.0) Everyone

eye_color_brown L:11% B:11% S:11% (wt:2.0) Everyone

eye_color_green L:17% B:17% S:17% (wt:3.0) Everyone

eye_color_hazel L:17% B:17% S:17% (wt:3.0) Everyone

eye_color_black L:6% B:6% S:6% (wt:1.0) Everyone

Hair Color

Add new item

hair_color_black L:19% B:19% S:19% (wt:20.0) Everyone

hair_color_brown_dark L:4% B:4% S:4% (wt:4.0) Everyone

hair_color_brown_darkash L:4% B:4% S:4% (wt:4.0) Everyone

hair_color_brown_darkintense L:4% B:4% S:4% (wt:4.0) Everyone

hair_color_brown_darkgolden L:4% B:4% S:4% (wt:4.0) Everyone

hair_color_brown_mediumash L:4% B:4% S:4% (wt:4.0) Everyone

hair_color_brown_mediumintense L:4% B:4% S:4% (wt:4.0) Everyone

hair_color_brown_light L:4% B:4% S:4% (wt:4.0) Everyone

hair_color_brown_mediumgolden L:4% B:4% S:4% (wt:4.0) Everyone

hair_color_brown_mediumcopper L:4% B:4% S:4% (wt:4.0) Everyone

hair_color_brown_lightcopper L:4% B:4% S:4% (wt:4.0) Everyone

hair_color_blonde_light L:7% B:7% S:7% (wt:8.0) Everyone

hair_color_blonde_medium L:10% B:10% S:10% (wt:11.0) Everyone

hair_color_blonde_mediumgolden L:11% B:11% S:11% (wt:12.0) Everyone

hair_color_blonde_verylight L:7% B:7% S:7% (wt:8.0) Everyone

hair_color_blonde_mediumash L:7% B:7% S:7% (wt:8.0) Everyone

Skin Color

Add new item

skin_color_pale L:7% B:7% S:7% (wt:5.0) Everyone

skin_color_mild_tan L:4% B:4% S:4% (wt:3.0) Everyone

sc_ca_01 L:15% B:15% S:15% (wt:10.0) Everyone

sc_ca_02 L:15% B:15% S:15% (wt:10.0) Everyone

sc_ca_03 L:15% B:15% S:15% (wt:10.0) Everyone

sc_ca_04 L:15% B:15% S:15% (wt:10.0) Everyone

sc_ca_05 L:15% B:15% S:15% (wt:10.0) Everyone

sc_ca_06 L:15% B:15% S:15% (wt:10.0) Everyone

Hair Style

Add new item

Short Wavy Quiff 01_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Short Straigh tQuiff 02_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Short Straight Mohawk 01_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Medium Wavy Wolfcut 03_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Medium Wavy Wolfcut 01_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Undercut 01_v01 None L:7% B:7% S:7% (wt:10.0) Everyone

Undercut 02_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Undercut 03_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Short Wavy Quiff 01_v01m L:7% B:7% S:7% (wt:10.0) Everyone

Short Wavy Drop Cut 01_v01

Short Wavy Drop Cut 01_v01m

Short Straigh tQuiff 02_v01m

Add new item

Medium Straight Ponytail 01_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Short Wavy Quiff 01_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Short Straight Mohawk 01_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Medium Braid Ponytail 01_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Medium Curly Ponytail 01_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Medium Wavy Wolfcut 03_v01 L:7% B:7% S:7% (wt:10.0) Everyone

Undercut 01_v01 None L:1% B:1% S:1% (wt:1.0) Everyone

Undercut 02_v01 L:1% B:1% S:1% (wt:1.0) Everyone

Undercut 03_v01 L:1% B:1% S:1% (wt:1.0) Everyone

Medium Straight Ponytail 01_v01m

Short Wavy Quiff 01_v01m

Short Wavy Drop Cut 01_v01m

File Edit Asset Window Tools Help

Default_01_GK

Raw Data | Goals Eth...y Editor

Boots

Add new item

shoes_01a_v01 — L:25% B:25% S:25% (wt:10.0) — Everyone

shoes_01b_v01 — L:25% B:25% S:25% (wt:10.0) — Everyone

shoes_01c_v01 — L:25% B:25% S:25% (wt:10.0) — Everyone

shoes_01d_v01 — L:25% B:25% S:25% (wt:10.0) — Everyone

Gloves

Add new item

gloves_03_v1 — L:25% B:25% S:25% (wt:10.0) — Everyone

gloves_03_v2 — L:25% B:25% S:25% (wt:10.0) — Everyone

gloves_03_v3 — L:25% B:25% S:25% (wt:10.0) — Everyone

gloves_03_v4 — L:25% B:25% S:25% (wt:10.0) — Everyone

Sleeves

Add new item

sleeves_01b_both_v01 — L:50% B:50% S:50% (wt:10.0) — Everyone

none — L:50% B:50% S:50% (wt:10.0) — Everyone

Jersey

Add new item

default01_gk — L:50% B:50% S:50% (wt:10.0) — Everyone

default01_longsleeve_gk — L:50% B:50% S:50% (wt:10.0) — Everyone

Shorts

Add new item

default01_gk — L:100% B:100% S:100% (wt:10.0) — Everyone

Socks

Weighted arrays

- Weighted array entries preserve real world distributions
- Randomly select an entry using the weighted sum
- Used for selecting ethnicity, as well as each visual feature

Weighted arrays snapshot

- Snapshot contains weighted array entries optimized for runtime
- Artist friendly tables are expanded into multiple weighted arrays
- Entries store the weighted sum of previous entries including self
- Last entry will therefore contain the weighted sum for all entries
- Select random value from 0 to weighted sum of all
- Use either linear or binary search to find entry represented by value

Weighted arrays code sample

```
static FName GetRandomWeightedName(const FWeightedArray& Array, const FRandomStream& Stream)
{
    const float SumWeight = Array[Array.Num() - 1].SumWeight;      // last element contains weighted sum of array
    const float RandomWeight = Stream.FRandRange(0.0f, SumWeight); // pick random value in range 0..SumWeight
    int i = LinearSearchWeightedArray(Array, RandomWeight);        // find element for RandomWeight
    return Array[i].Name;
}

static int LinearSearchWeightedArray(const FWeightedArray& Array, float RandomWeight)
{
    for (int i = 0; i < Array.Num(); i++)
    {
       if (RandomWeight <= Array[i].SumWeight) // return first element that includes RandomWeight
          return i;
    }
    return Array.Num() - 1;
}
```

Deterministic randomness

- Must use random number generator deterministically
- Always call random for visual features in same order
- Must allow artists to add visual features over time
- Would like to have characters evolve as their rating increase

Snapshot and versioning

- For each character content release we perform a snapshot
- Snapshots contain the order visual features are calculated
- On character creation server stores seed along with latest snapshot version

Snapshot contains the following tables

- Nationality to ethnicity mapping
- Ethnicity/gender group to visual string mapping

Evolving characters

1) Populate all features as normal

a) Hidden features (tattoos/ear-rings)

1) Shuffle order of hidden features from character seed

2) Remove features that remain hidden for evolution tier

b) Evolving features (hair style)

2) Artists describe evolution chains for values

3) Use feature value as base value

4) Redetermine feature value for each evolution tier

Blueprints

- Blueprints hide how strings are translated to visual data
- Provides a common interface for all visual features
- Makes it easy to support new visual features with existing tools
- 16 feature handlers
- Used for body parts and clothing

Engine Subsystem

- Character generation is implemented as an engine subsystem
- We use Unreal's FTask for multithreading
- Subsystem caches tasks of recently generated characters
- Hash of input string map is used for cache key
- Avoids generating the same character multiple times
- Allows result of task to easily be used with multiple task chains

Runtime asset composition

- Copy mesh data
- Morph mesh pieces for gender/build
- Additive adjustments to facial features
- Clip body mesh hidden by clothing
- Add render sections for decals
- Merge mesh pieces

Morph mesh pieces

- Interpolate vertex data from base mesh to morph target
- Used to support gender and bulkiness
- All bodies are based of the same base skeletal mesh
- We use character’s BMI to determine interpolation value
- Tooling allows artists to easily provide additional morphs for clothing

Facial features

- Base head is selected from ethnicity
- We use additive adjustments to make faces unique
- Adjustments are achieved by modifying facial bones
- Artists provide relative min/max transformations per facial feature
- Each base head can have custom min/max limits
- We interpolate transformations using random value per feature
- We apply the resulting transformation using CPU skinning

Preview...Settings | Card Preview | Facial Features | Characte...positor

Search

Nose	Random
nose_profile	-0.015761
nose_forward	0.29687
nose_height	-0.733371
nose_length	0.646314
nose_nostril_height	0.063175
nose_nostril_width	-0.905741
nose_ridge_angle	-0.017439
nose_ridge_width	0.082362
nose_size	-0.832096
nose_tip_angle	0.394659
nose_tip_forward	-0.777849
nose_tip_width	0.596615
Eyes	Random
eye_angle	0.946873
eye_depth	0.026095
eye_distance	-0.397305
eye_height	0.561198
eye_shut	-0.263492
eye_size	0.403968
Ears	Random
ear_angle	0.612654
ear_bend	-0.656553
ear_outward	-0.107135
ear_size	0.921156
Cheeks	Random
cheek_forward	0.239014
cheek_height	-0.525152

Quick side note

- Originally string map population occurred on server
- Avoided snapshots as cloud stored each character string map
- Changes to character generation had to involve cloud team
- This included art content updates, slowing iteration time
- Facial features was the breaking straw that convinced us to move client side

Decals

- Artists are free to add decals to clothing
- Artists specify placement of decal in UV space
- Any triangle that intersects with decal placement will be added to new render section
- Split existing index buffer by moving intersecting triangles to end

Decals code sample

```
uint32 BaseIndex = RenderSection.BaseIndex;
uint32 EndIndex = BaseIndex + RenderSection.NumTriangles * 3;
while (BaseIndex < EndIndex)
{
    bool ContainsAny = false;
    for (int i = 0; i < 3 && !ContainsAny; i++)
    {
        const uint32 VertexIndex = LODRenderData.IndexBuffer[BaseIndex + i];
        FVector2f UV = LODRenderData.GetUV(VertexIndex, UVIndex);
        FVector2f TransformedUV = UVTransform.TransformPoint(UV);
        ContainsAny = UnitBox.IsInsideOrOn(TransformedUV);
    }
    if (ContainsAny)
    {
        EndIndex -= 3;
        for (int i = 0; i < 3; i++)
        {
            std::swap(LODRenderData.IndexBuffer[BaseIndex + i], LODRenderData.IndexBuffer[EndIndex + i]);
        }
    }
    else
    {
        BaseIndex += 3;
    }
}
```

Clipping

- Remove triangles contained within clipping mesh
- Test if ray from vertex hits clipping mesh an odd or even amount
- Use an acceleration grid to reduce ray intersection tests
- Assume triangles are evenly distributed within the clipping mesh bounding box
- Project along the shortest axis of bounding box
- Find optimal cell size that gives us desired triangle count per cell
- Store clipping mesh triangles per grid cell

Clipping code sample

```
ClippingMeshAccelerationGrid AccelerationGrid = GenerateClippingMeshAccelerationGrid(ClippingMesh);

FVector3f Direction = FVector3f::ZeroVector;
Direction[AccelerationGrid.ProjectedAxis] = 1.0f;
UE::Geometry::FIntrRay3Triangle3f Intersector(FRay3f(FVector3f::ZeroVector, Direction), UE::Geometry::FTriangle3f());
for (int i = 0; i < LODRenderData.PositionVertexBuffer.Num(); i++)
{
    // Ignore vertices outside AABB
    const FVector3f& Point = LODRenderData.PositionVertexBuffer[i];
    if (!FMath::PointBoxIntersection(Point, AccelerationGrid.AABBox))
    {
       continue;
    }

    Intersector.Ray.Origin = Point;
    const FIntPoint IntersectingCell = AccelerationGrid.GetCellPoint(Point);
    if (IsInClippingMesh(Intersector, ClippingGeometry, AccelerationGrid.GetCell(IntersectingCell)))
    {
       OverlappingVertices[i] = true;
    }
}
```

Merge mesh pieces

- Based of Unreal’s SkeletalMeshMerge
- Combines render sections referencing the same material

Your done!

- That’s an overview of our character generation pipeline
- We have just generated a character from a seed to be used in UE5

Thanks to my colleagues

- Torbjörn Söderman
- David Serrat Jiménez
- Andrei Kushner
- Nikolaos Kaltsogiannis
- Marco Musto
- Albin Lundahl
- Daniel Noll

Bonus slides

Texture baking

- Optimization for low detail characters
- Merge render sections with generated texture atlas
- Texture atlas is BC7 encoded using compute shader
- Compute shader writes encoded block to R32G32B32A32_UINT
- FComputeShaderUtils::AddCopyTexturePass does not allow copying between different pixel formats