ExoGP Level Design Workflow Documentation

Big Moxi Games Level Design Team

Introduction:

This document will explain the Level Design (LD) Workflow that the LD team developed to be able to build Tracks for the ExoGP in the most efficient way while still maintaining the core functionality and narrative of the game.

We understand that ExoGP is a Flying Racing Game and the tracks that will be part of it draw inspiration from Formula 1 Racing tracks with well-defined Curves and Straights but adding an element of verticality and freedom.

Glossary:

Segment: A piece of the Track, the whole track is made up of many segments, each one with its number that increases during the length of the track. Straights and Curves are separated as segments and joined later on the process;

Curves: The Curves are defined by their **Radius** and **Angle** and can vary on those parameters.They can also be part of Ascents or Descents;

Radius: The Radius of the curve,measured from a point at the center of the arch;

Angle: The angle of the arch;

Straight: A straight segment of track, can be **short**, **medium** or **long**.Doesn’t need to be perfectly straight throughout the whole segment. A Stretch of track that is **under 10º** of inclination in any direction is considered a straight;

Ascents: Moments where the track **gains altitude, increasing its Z axis**;

Descents: Moments where the Track **loses altitude, decreasing its Z axis**;

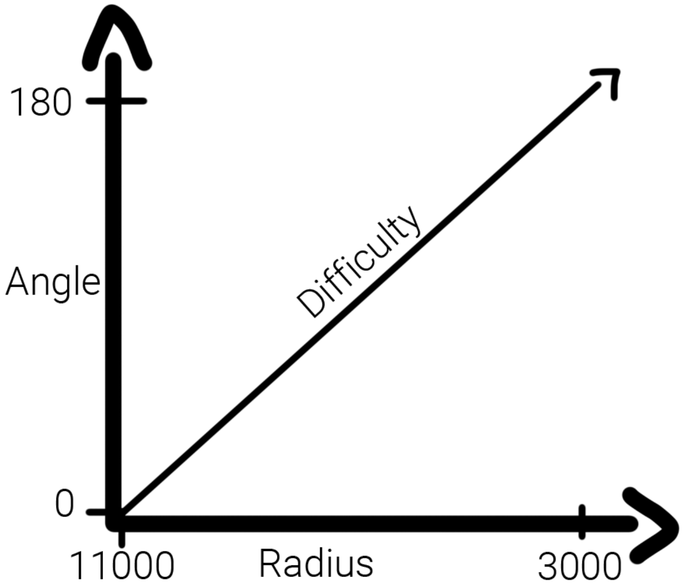
Chicanes: A **macro segment** of track composed of many smaller segments with the intent of giving a Zig-Zag motion to the track.

- Tools:

  - PureRef: Reference Collecting
  - Photoshop: Sketch / Track Planning
  - 3Ds MAX: Spline Making
  - Unreal Engine: Blocking and gameplay testing

- Metrics:

  - Curves:
    - The way curves are measured is based on their **difficulty level**. The difficulty depends on the speed that the turn can be made without crashing into the checkpoint.
    - We tested curves that range from radius 3000 to 11000 and came to the conclusion that the **smaller the radius, the harder it is** to make the turn. The angle of the curve also affects its difficulty, the **higher the angle, the harder it is**.



        - We found that angles that are multiples of 5 give the track a cleaner look so we suggest that all of the angles end with either 5 or 0 with the exception of Ascents and Descents.

- ### How we measured difficulty:

  The numbers you will see below are all derived from a formula that according to testing has a 3 to 5% margin of error.We attribute that to human factors since we used the average velocity of each turn to validate the numbers.

  Firstly we tested each curve and recorded the **average speed** and **time** it took the player to make the maneuver. With these data we could check if the metrics we came up with would hold and make sense on a bigger scale.

  The following formula describes a number that will inform how difficult a maneuver is, the range depends on the curves we have. For example, the hardest curve we have at the moment is the 3000r - 180º.

  $$\left(\frac{Angle}{Radius}\right) \times 100$$

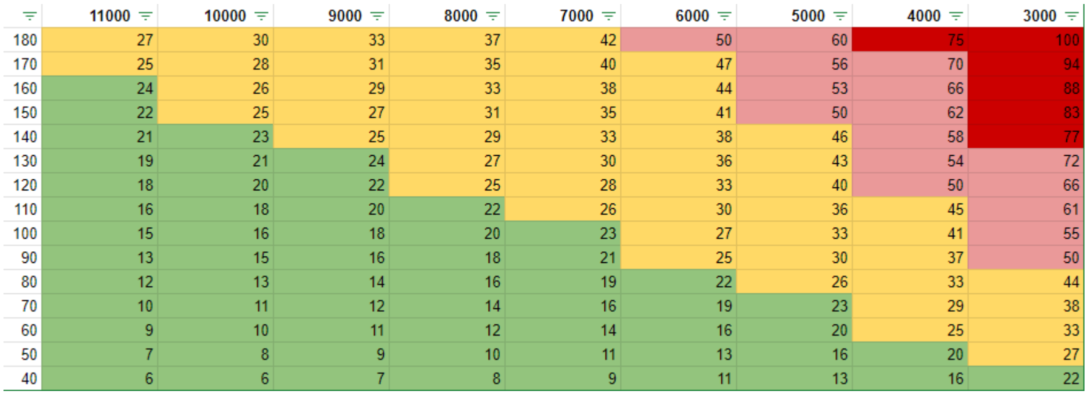
  The "*100*" is there just so that the numbers aren't too low and easier to manage.

  At the moment, this equation will give us a number that ranges from 0 to 6, **this range can *increase* if either the *angles increase* or if the *radius decreases* beyond the 180º or below the 3000r mark respectively**.

  Now that we have a number that defines each maneuver, we can normalize it so it is easier to understand and calculate with. We defined a range that goes from 0 to 100 translating 6 to 100, 3 to 50, 0 to 0, etc... Where 0 is basically a straight line and 100 is the hardest curve the game has at the moment.

  We decided to separate the curves in 4 categories based on their difficulties: Easy, Medium, Hard and Very Hard.

Here are all of the curves that we tested, on the **columns represents the Radiuses that *decrease* from left to right** and on the **lines are the Angles that *decrease* from top to bottom**.The numbers on each cell are the difficulties for each curve.



	11000	10000	9000	8000	7000	6000	5000	4000	3000
180	27	30	33	37	42	50	60	75	100
170	25	28	31	35	40	47	56	70	94
160	24	26	29	33	38	44	53	66	88
150	22	25	27	31	35	41	50	62	83
140	21	23	25	29	33	38	46	58	77
130	19	21	24	27	30	36	43	54	72
120	18	20	22	25	28	33	40	50	66
110	16	18	20	22	26	30	36	45	61
100	15	16	18	20	23	27	33	41	55
90	13	15	16	18	21	25	30	37	50
80	12	13	14	16	19	22	26	33	44
70	10	11	12	14	16	19	23	29	38
60	9	10	11	12	14	16	20	25	33
50	7	8	9	10	11	13	16	20	27
40	6	6	7	8	9	11	13	16	22

Easy Curves:GREEN 45,19% of total curves (0 <= x < 25)

Curves that can be used throughout the track to connect sectors or to give the player some breathing space while transitioning from one segment to another.

Medium Curves:YELLOW 38,52% of total curves (25 <= x < 50)

Curves that are a bit harder to make, require the player to either slow down or to enter at just the right angle at higher speeds.Should be used during chicanes or changes of elevation. Gives the player a choice between slowing down or speeding up at risk.

Hard Curves:ROSE 11,85% of total curves (50 <= x < 75)

The player will need to slow down and focus to be able to make this turn. Should be used with care to not make the track too hard or confusing. Can be used during Ascends or Descents to force the player to really slow their speed.

Very Hard Curves:RED 4,44% of total curves (75 <= x <= 100)

The player must slow down or they will crash into something. Should be used after a long Straightway to change the pacing of the track. Must be used only in moments where it's imperative that the player slows down. Shouldn't be the first option to be used during Ascends or Descents because it gets a bit confusing and adds an element of difficulty that we have better options to use instead, like Hard or Medium Curves.

- Curves Composition:

  One aspect that we tested and are happy to report is that this measuring system also works when adding two or more curves together. For example, the ***5000r-120º*** **curve has a difficulty level of** ***40*** and the ***11000r-90º*** **curve has a difficulty level of** ***13***. **If we add three** ***11000-90º*** **curves in** ***sequence*** **that would add to** ***39 collective difficulty*** and the final average speed of this 3 curve segment is similar to the one that has a difficulty of 40.

  We are accepting a fluctuation of 5% at most, due to the nature of the test being human made.In this case to match the one 40 difficulty curve, the result of the 3-curve composition would accept a range of 38 to 42.

- Ascents and Descents:

  There is the matter of **Ascents** and **Descents** that each pose its own difficulties to the track. According to the tests we ran here are what we could observe:

  Ascents: Due to **gravity**, the player **loses speed when going upward**.While in one hand this **makes the curve easier** because the **brake action is stronger** and therefore the player can **enter ascents segments with higher speeds**, on the other, the decrease in speed makes it seems like it is a harder maneuver to make than it actually is.For that reason and based on the tests we suggest **reducing 5% of the difficulty for a segment that is an ascent.**

  The opposite is true to Descents as well. Due to **gravity**, the player **gains speed while going downward** making the **braking action less effective** and forcing the player to either **start braking early or entering the curve on a much riskier angle**. For those reasons and based on the testing we suggest **adding a 5% difficulty to a segment that is a Descent**.

- Straights:
  - Straights are moments of the track that allows the players to gain speed.These moments don’t necessarily need to be perfectly straight, it can have a small curvature to it so that it is more

interesting. Straights are also segments that allow for some decision making like an obstacle that blocks the path forcing the player to go to either side or risking going through the middle and gaining an advantage.

Short Straights

Usually these shorter segments connect other segments such as curves or at the beginning and end of an ascent or descent.

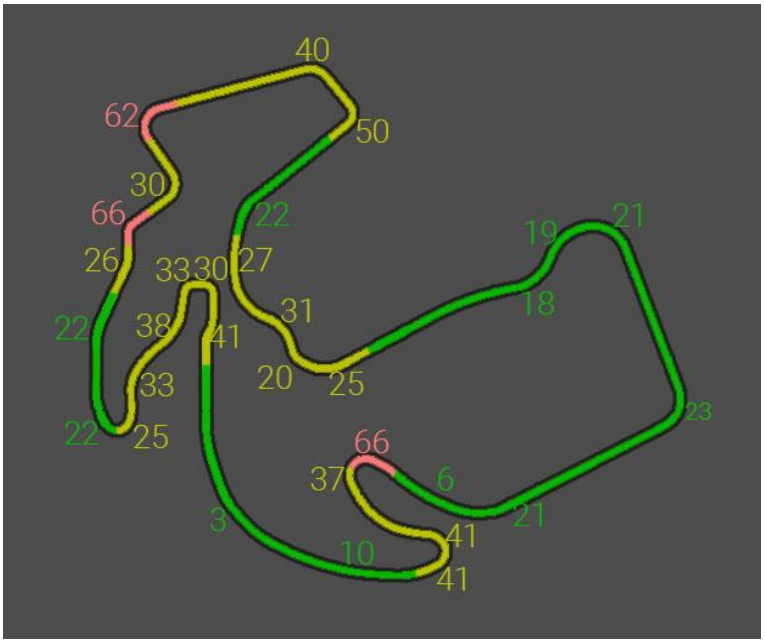
Medium Straights

Medium segments that allow for moderate speed and decision making, connecting sectors of the track and before or after chicanes.These can be used to maintain the tension and prepare the player for another sector of the track.

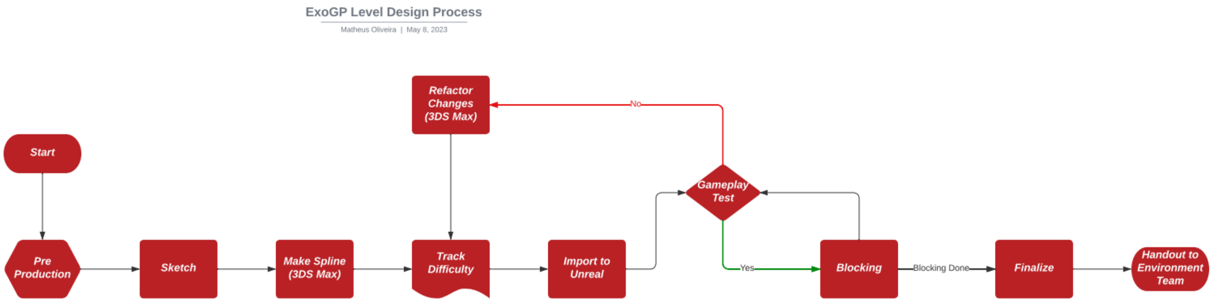
Long Straights

Here is where the speed happens.Every track should have at least one for the Start/End of the Track. Longer paths allow for much more complex decision making moments where the player should choose to speed up or to fight their opponents.

TestTrack Example



Workflow:



- 1 - Concept/Sketch. Collect References and Make a **Sketch** with the outline of the Track as **viewed from the top**.
    - 1.1 - Collect References and brainstorm ideas.
    - 1.2 - At this point, the obligatory moments and general shape of the Track are defined roughly on the Sketch.
    - 1.3 - Make the Sketch.

- 2 - **Make the Track**(Spline) on 3DS Max.
    - 2.1 - Make the Spline from the Top View. At this point, the track **is still Flat**, no Ascents or descents.
    - 2.1 - Decide where to place the ascents and descents.
    - 2.2 - Make the ascents and descents on the Track.
    - 2.3 - Separate the track in segments and derive the overall difficulty of the track.
    - 2.4 - Finalize the Track on 3DS Max making sure all the points are connected and that the Track is both looking good, interesting to play and has space for the moments decided on Step 1.
    - 2.5 - Derive Track Difficulty
    - 2.6 - Import the Track into Unreal Engine.

- 3 - With the Track inside Unreal, we can start to connect the parts that **Make the Track Playable**.
    - 3.1 - Connect the TrackMaker
        - 3.1.2 - Cleanup the Track so that it is smooth and with nothing blocking the player.
    - 3.2 - Connect the CheckpointMaker
        - 3.2.1 - Decide roughly where to place the checkpoints, this is just the first placement, it will be fine-tuned during Step 4.
    - 3.3 - Connect the PathIndicator

        - 3.3.1 - Make sure that the indicator doesn’t have any sharp corners due to the spline positioning.
    - 3.4 - Track Cleanup. Making sure that the track boundaries are not clipping on itself.
-
- At this point the Track is **ready to play and test** if everything is as it should.

- At this point is where we can See if the Track is actually **fun to Play** and we can make **minor changes inside Unreal** Engine as Needed.

- **Bigger Changes should go Back to Step 2**, But would take less time.

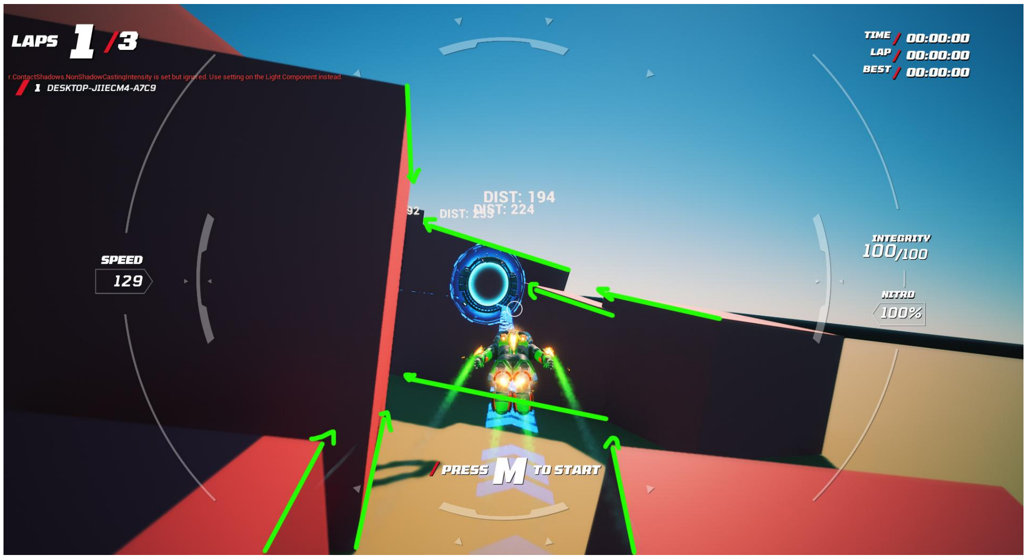
- 4 - Now that we have a Track that is working and fun to play, we can **Start the Blocking Process** on Unreal.

- 4.1 - Define the Rough shape of the Landscape.
- 4.2 - Block Natural environment and Man Made Structures that will affect the Track.
- 4.3 - Block Line of Sight from one Checkpoint to the others that aren't the Immediate Next one.
    - 4.3.1 - Fine-tune the positioning of Checkpoints.
- 4.4 - Playtest as we go along Step 4

- 5 - Finalize the track, double-check everything and hand it to the environment team.

- Level Design Insights:

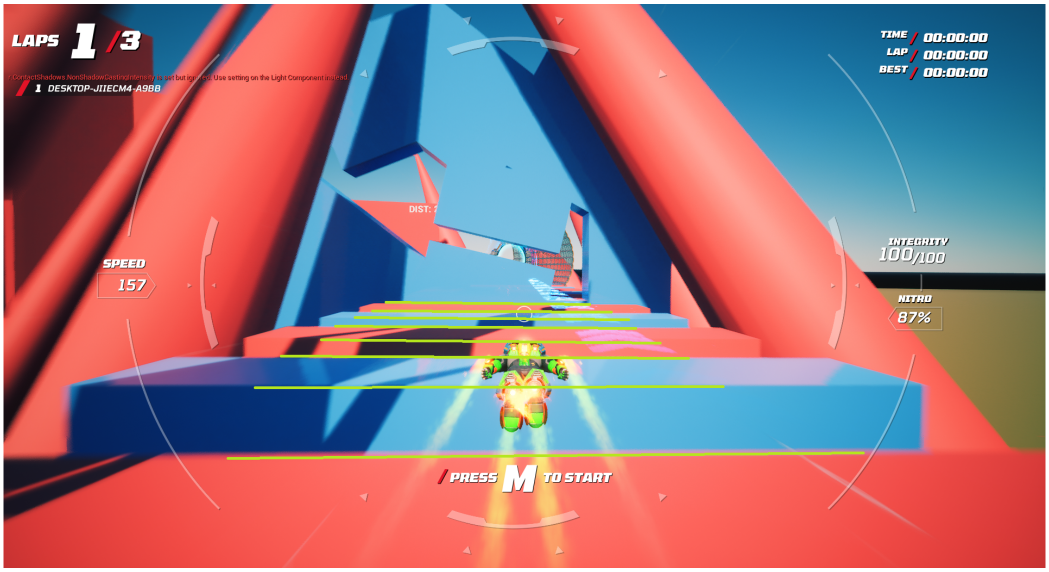
Leading Lines: It is important that the player knows where to go at all times and doesn’t get lost, we should use the edges of the blocks that make up the map to inform the player where to go next.



r.ContactShadows.NonShadowCastingIntensity is set but ignored. Use setting on the Light Component instead.

Framing: Using blocks to create frames that look nice and inform the player on where to go next. This is similar to the leading lines, we should block the player's vision and make special places on the track that focus the players attention on important parts of the map.

Sense of Speed: This is one of the most Important factors when evaluating how fun a racing game is.There should always be something on the player peripheral vision that is coming towards the camera. Vertical and horizontal lines are the most effective way of conveying this sense of speed. It's also possible to create these lines using light and shadow. During areas that are too tight or don’t have many of these lines such as tunnels, we can add lots of smaller elements to keep the sense of speed high.



r.ContactShadows.NonShadowCastingIntensity is set but ignored. Use setting on the Light Component instead.

Pacing: Mixing Straights and curves in a manner that is fun and interesting is important so that the track doesn't get boring or too difficult. We also divided the track in sectors, giving each a personality and features that are unique to it. For example, the Canyons Sector on the Obelisk Track will feel different from the Obelisk Sector and the Pleasance Station Sector.

Duration: Minimum of 01:00 to a maximum of 03:00 minutes per Lap (for now). Once combat is implemented, this time might change since the player's attention will be divided between Racing and Combat and so the feeling of how long the track is could be diluted, therefore increasing the time.

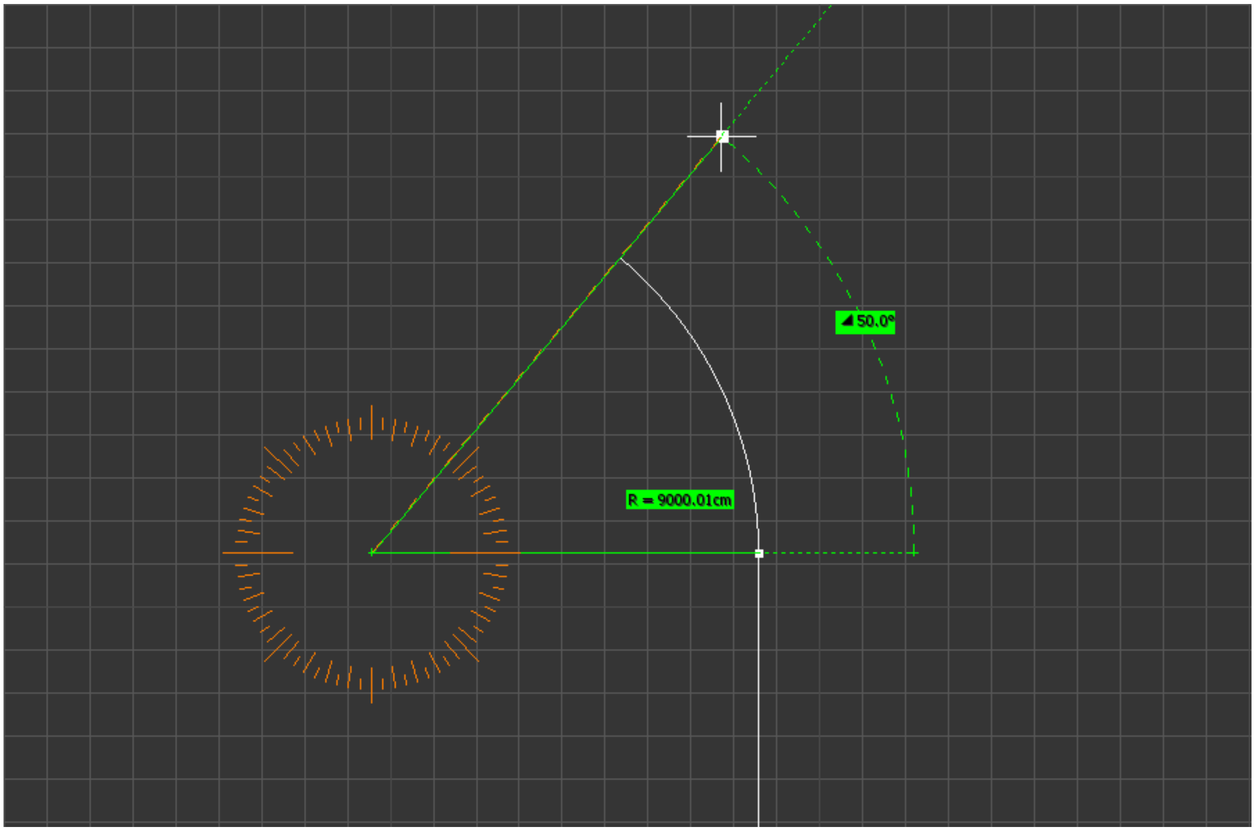
Complexity: Increasingly complex tracks throughout the biome, using the tracks to teach maneuvers like roll and give the player a feeling for the controllers. We could use the sum of all the maneuvers to get an estimate of how difficult the track is and design accordingly, increasing the difficulty as the game progresses.

- About the Tools:

We are using a script within 3Ds MAX that allows us to have precise control of the angles and radius of each maneuver. This script is already part of the workflow and we found a way to make the tracks in a manner that is straightforward once the sketch is approved.

Inside the 3ds Max we can also normalize the amount of points the spline will have once imported into unreal. This helps to solve issues that we had with the tangents.

Example of the script functionality



This functionality of displaying the Angle and Radius should be possible to implement inside Unreal but we find that it's not worth it since we already have the tools needed at our disposal. There are also the issues with the distribution of points along the spline which would be harder to implement in Unreal and it’s already a basic functionality of 3ds Max and works well for our purposes. Also the cost of making this tool inside Unreal is high in comparison to the Max tool.