TicTacToeFlight (V1.0)

Flight created on 10.01.2025

Estimated flight duration 15-30 min.

Difficulty level: Easy

Task: Play Tic Tac Toe with any airplane.

Introduction:

Lockheed Martin has included a “Tic-Tac-Toe Script Example” with the flight simulator.

This example is

- graphically not very appealing,
- you play against yourself (no computer opponent) and
- you simply stand on the runway while playing.

That's why I wanted to find out whether Tic Tac Toe can also be

- in “nice”,
- with an “AI” as opponent and
- the whole thing as a flight.

I had a few problems to solve:

1) I can't script, so the game has to do without scripts. Is it possible to realize the project with actions, areas, triggers and objects only?

2) How do I create the playing field and the game pieces without creating additional scenario objects, which have to be installed separately for the mission?

3) How do I find out which field is occupied by the player and which by the AI?

4) How do I find out whether the player or the AI has won?

5) How do I tell the AI that it is its turn?

6) How do I “program” an (albeit simple) AI, using only actions, areas, triggers and objects?

7) How do I program an intelligent AI? There are many different game situations that the AI has to recognize / query in order not to lose.

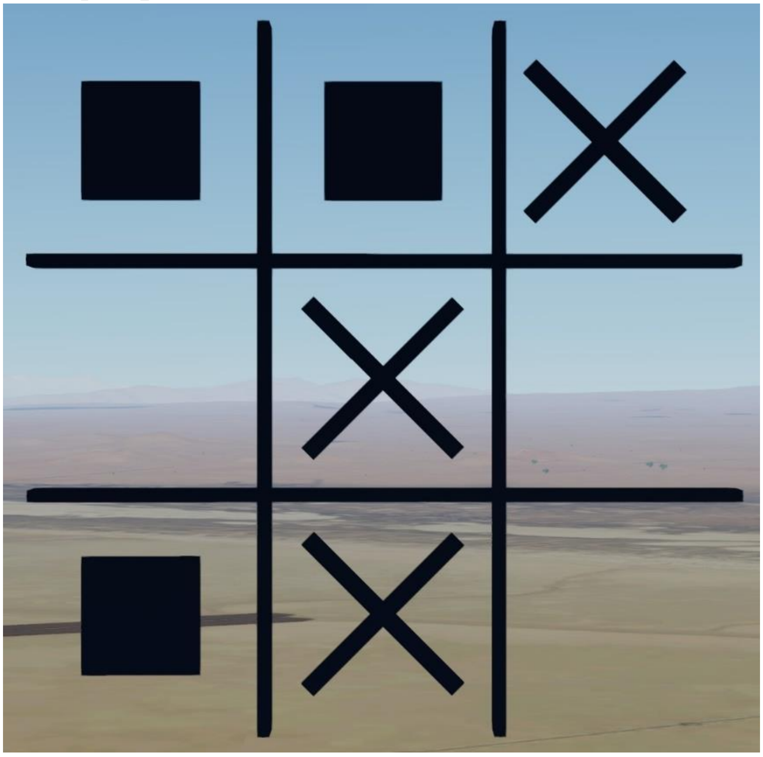
8) How do I differentiate between simple and intelligent AI after starting the game?

If you are interested, you can read my solution to the problems below.

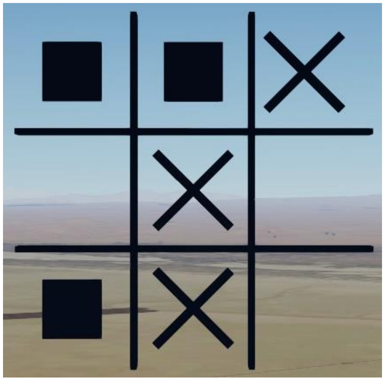
The playing field:

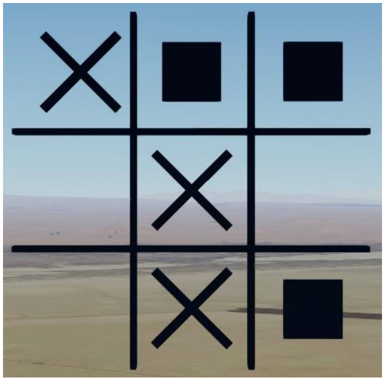
Everyone should be familiar with how Tic Tac Toe is played.

The player's token is an X, the AI's is a ■.



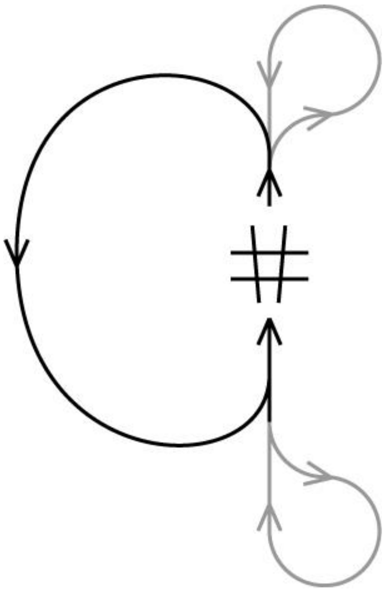
This flight - Tic Tac Toe has a special feature:
You can see the playing field from two sides.





From the other side, all the tiles already placed are mirror-inverted.
So you have to “rethink” after turning them over.

You can fly through the playing field from both sides, here are two possible flight routes:

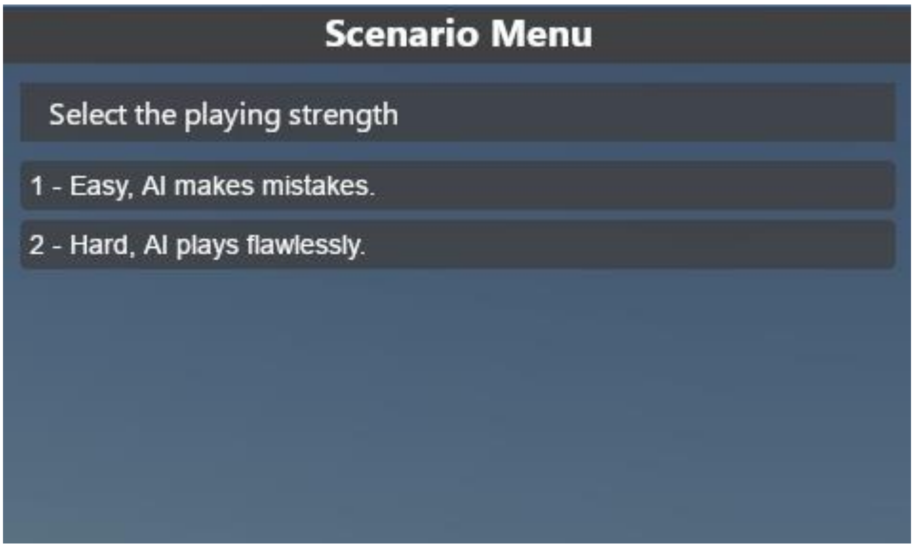


1) Fly one lap around the playing field and always approach from the same side.

2) Turn 180° after each pass and approach the field again.

The start of the game:

1)At the beginning, you choose the AI's playing strength.
In "Easy" mode you can win, in "Hard" mode the maximum possible score is a draw.



Scenario Menu

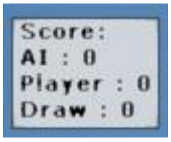
Select the playing strength

1 - Easy, AI makes mistakes.

2 - Hard, AI plays flawlessly.

2) You then have 10 seconds to change aircraft.
Choose any aircraft, helicopters and gliders are also allowed (see below).

3) The game lasts 5 rounds, the first player to occupy 3 squares in a row, column or diagonal wins a round.
A display at the top right shows the current score.



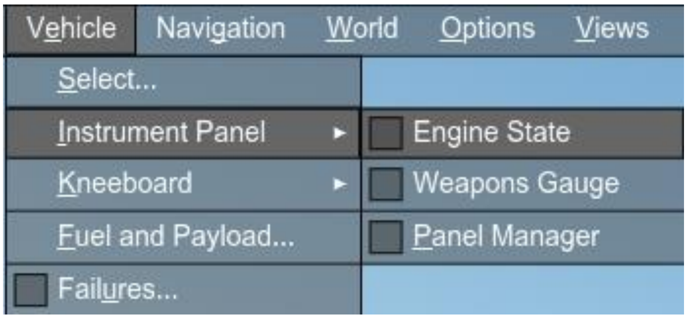
Score:
AI : 0
Player : 0
Draw : 0

4) Do not touch the field lines.
The collision does not damage the airplane, but the round is lost if you touch it.
That's why you have to approach very precisely with large airplanes. An outside view can be helpful.

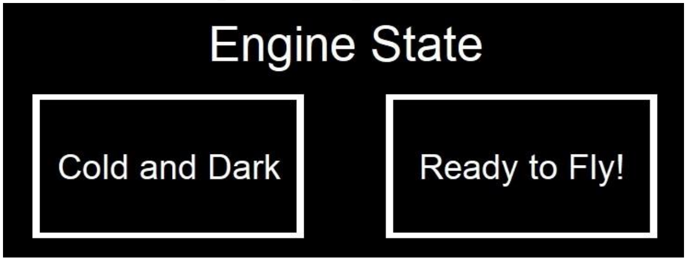
5) The player begins the first round. Start and fly through any field for the first turn.

The choice of aircraft:

F16:
If you want to stay with the F16, you must make the machine ready for take-off.
To do this, select Vehicle/InstrumentPanel/EngineState in the menu:



Choose Ready to Fly!



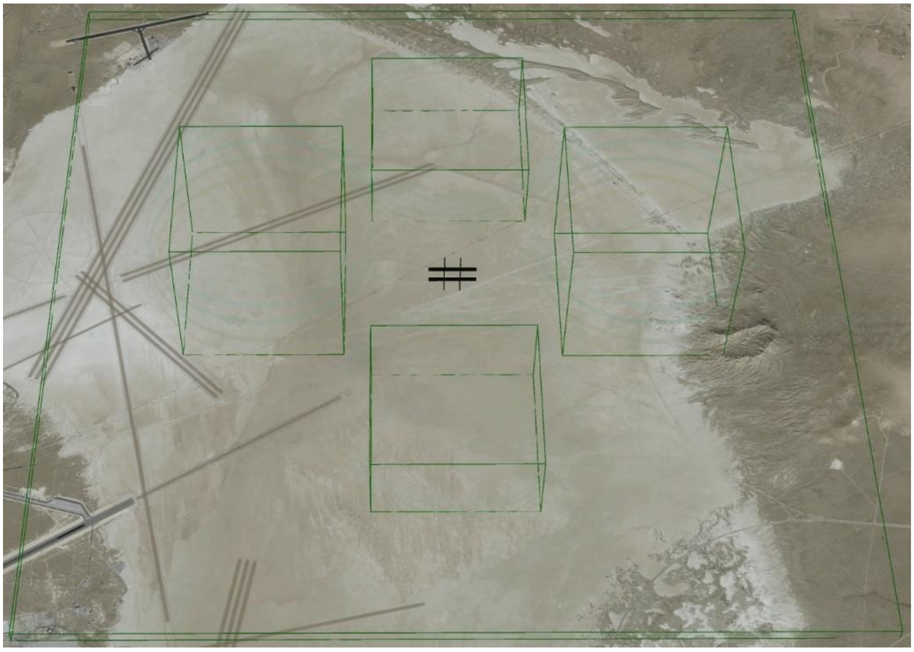
Glider:
When a glider is selected, it automatically recognizes the game (on the missing engine) and activates thermals.
(The thermals are not activated in gliders with an auxiliary engine, so choose a "pure glider" first, after activating the thermals you can also switch to a glider with an auxiliary engine).

To take off with a glider, you must request a tow plane with Shift+Ctrl+Y.
Let yourself be towed to a suitable height and release from the tow plane with Shift+Y.

You don't have to worry about running out of updrafts:
- The entire area around the field is covered by a base thermal, which extends up to 500 feet above the ground.

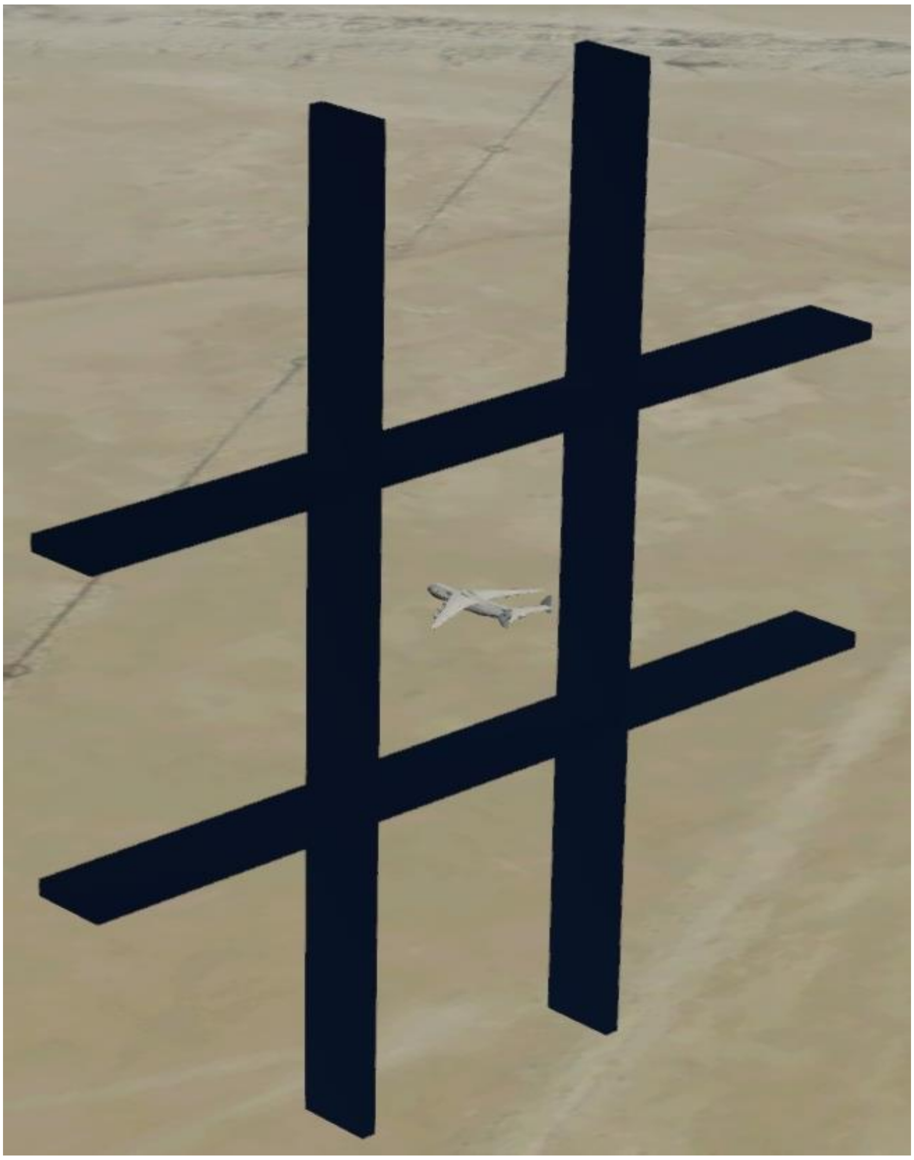
- There are thermals in front of and behind the field, with medium-strong updrafts.
- To the left and right of the field are thermals with strong updrafts.

Shown here with green frames for clarity.



You will recognize the thermals by flocks of birds.

Large airplanes:
The playing field is large enough for you to play with the largest aircraft, the AN225.

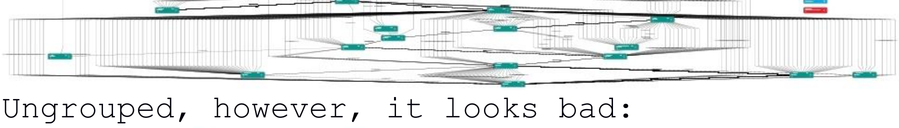


How I created the game:

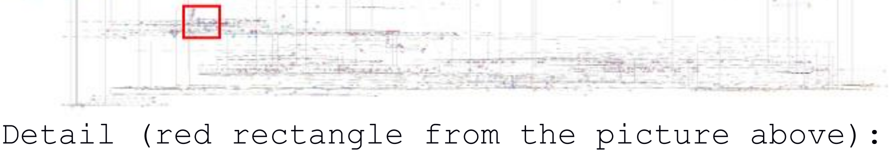
1) I can't script, so the game has to do without scripts. Is it possible to realize the project with actions, areas, triggers and objects only?

The whole game is really created without scripts. Only 944 objects were necessary, I have already created missions with 2000 objects. However, in Tic Tac Toe there are many more links between the individual objects than usual.

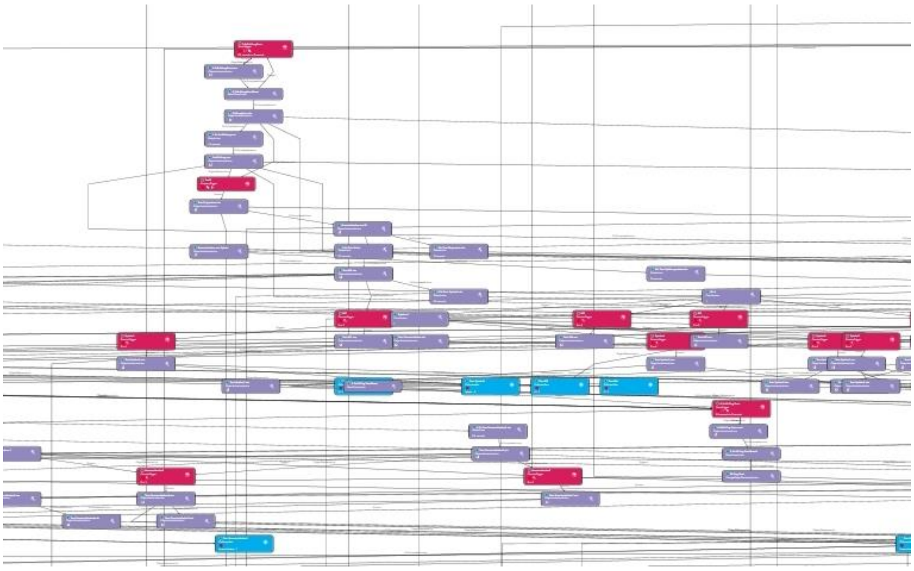
Grouped together, it is still reasonably clear:



Ungrouped, however, it looks bad:

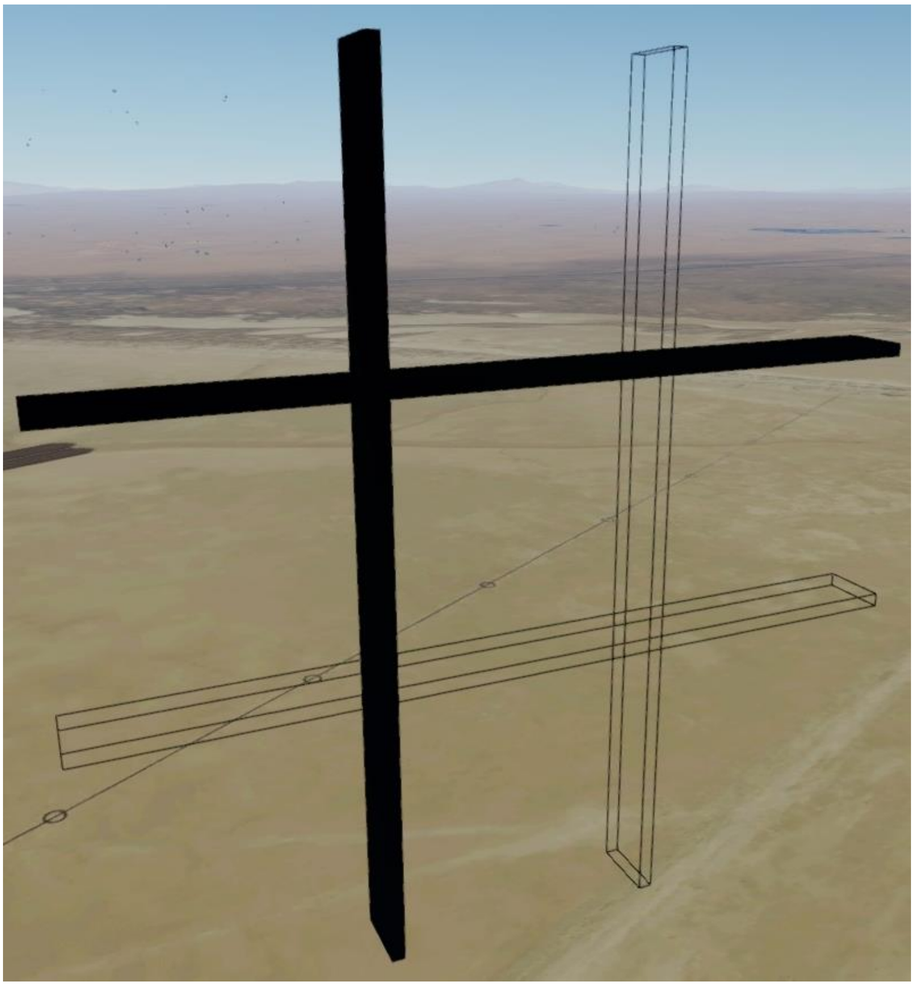


Detail (red rectangle from the picture above):



2) How do I create the playing field and the game pieces without creating additional scenario objects that have to be installed separately for the mission?

I simply created the playing field with color-filled areas. To illustrate this, here are 2 bars with outlines only.

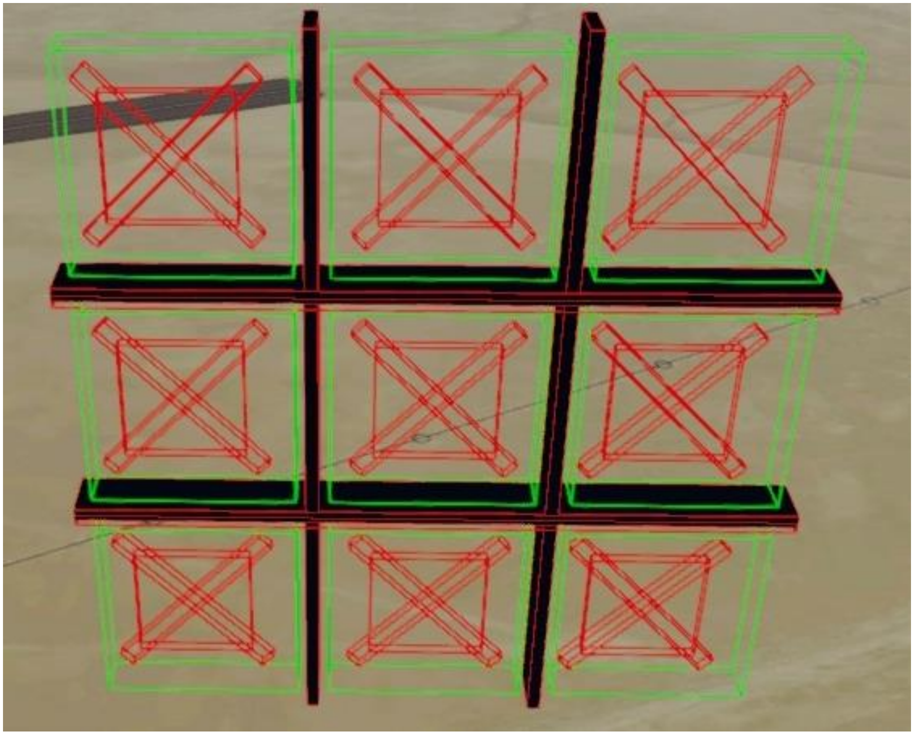


The game pieces are created in the same way.

2 areas are used for an X.

So that the player can understand which move the AI is currently making, the ■ is displayed “flashing” (white, gray, black).

One problem since P3dV5 is that I can't make the areas “invisible” by setting the color to transparent; a wafer-thin outline always remains visible. Only colorless areas are really invisible. So I placed colorless areas in the playing field (shown in red in the image below). Using ChangeObjectPlacementAction, a token that I have sunk (hidden) in the ground is moved to the position of the colorless area if required. At the end of a round, the tiles placed in the playing field are removed, i.e. hidden in the ground again.

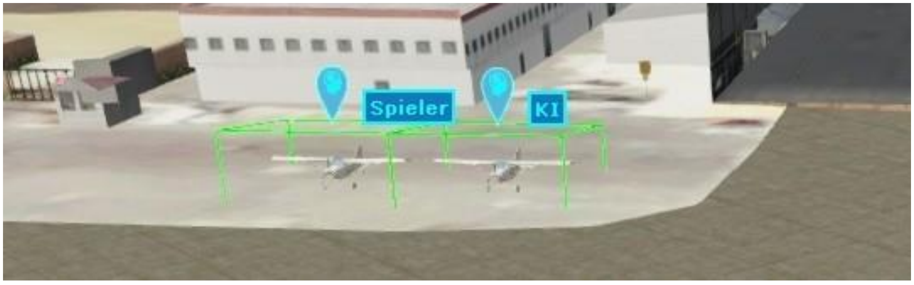


The areas shown in green are colorless and are used to query which field the player is flying through using a proximity trigger. As soon as the proximity trigger is triggered, the area is removed and replaced by another area, which then triggers a text when flying through (The field is already occupied). Because the AI does not fly through a field for a turn, this area must also be replaced when activating a ■.
Sounds complicated? It is, at least if you have to develop it yourself.

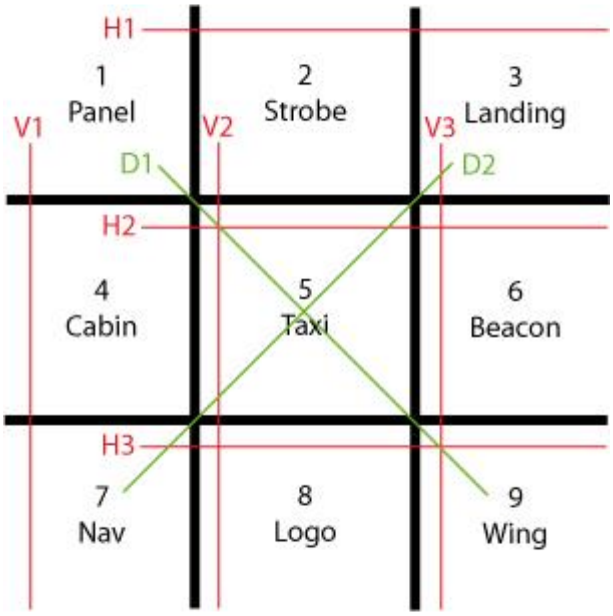
3) How do I find out which field is occupied by the player and which by the AI?
So far I only know which field is occupied, but not yet by whom. That was a big problem, because I can query this with PropertyTrigger, but I need 2x9 parameters that I can change and query. At first I thought of airplane lights, but different airplanes have different (or not all) lights, which I need. Also, there are no 18 lights, other parameters are hard to find and even if there were, the player could win by activating panel, strobe

and landing.

Finally I had the solution:
You can not only manipulate the lights of the player plane, but also that of each AI plane.
So I placed 2 AI airplanes, which are available in all P3d versions (4/5/6), at the airport.
Each playing field is assigned the same light for both aircraft.
Now I activate the corresponding light for the respective aircraft as required (using SetPropertyAction).



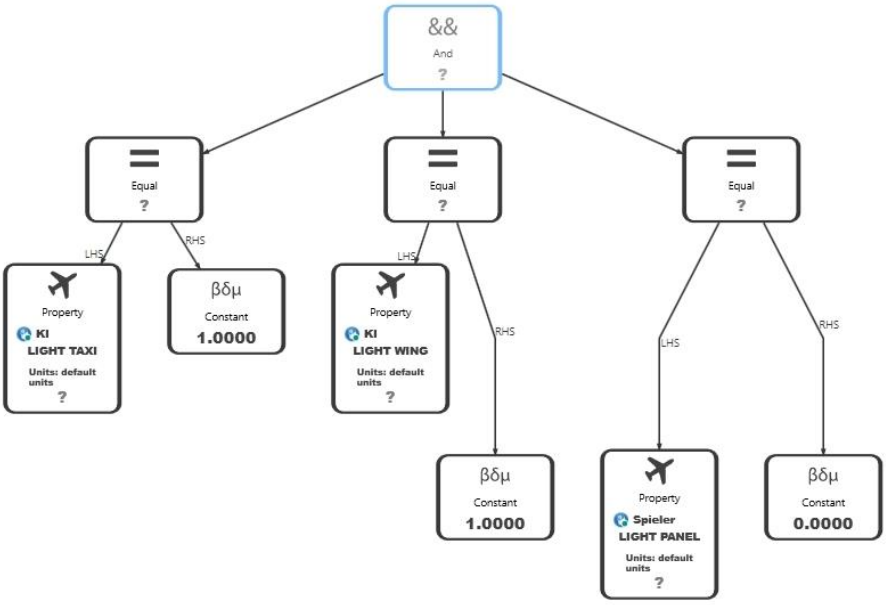
Here is the allocation of the lights:



1 Panel	2 Strobe	3 Landing
4 Cabin	5 Taxi	6 Beacon
7 Nav	8 Logo	9 Wing

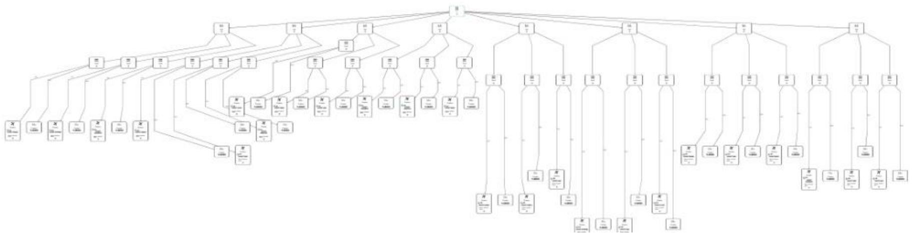
I now use PropertyTrigger to query which fields are occupied by whom.

In this example, the logic queries whether in the diagonal (from top left to bottom right), fields 5 and 9 are occupied by the AI and field 1 is not occupied by the player. If this is the case, the AI places its token in field 1 and wins.

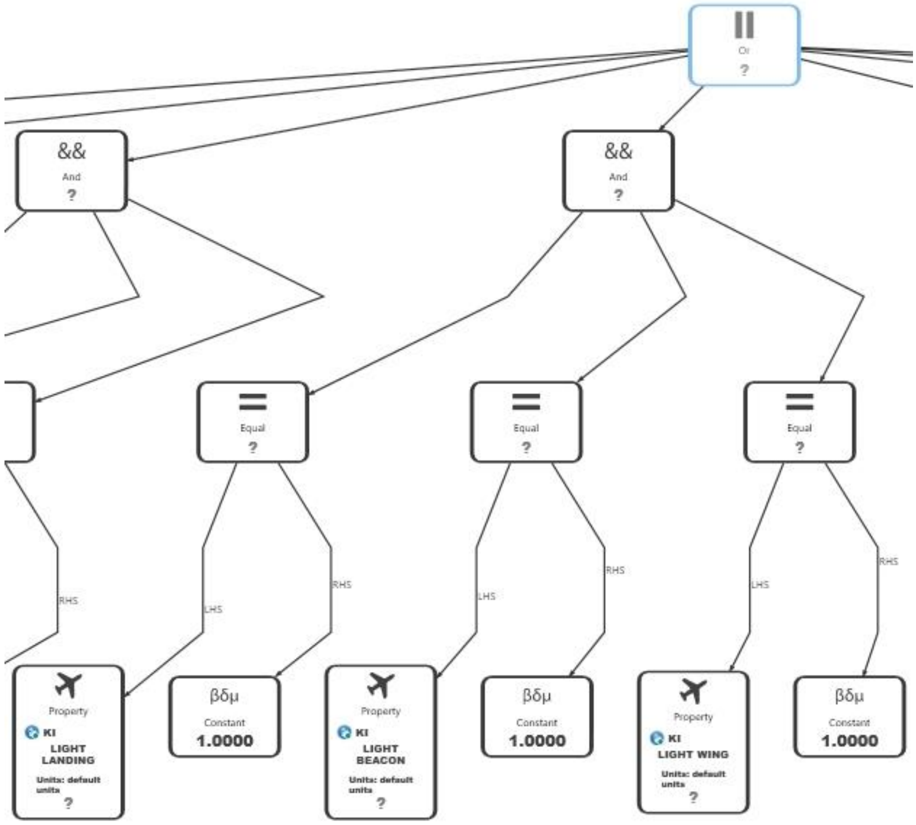


4) How do I query whether the player or the AI has won?

As in the example above, I use PropertyTrigger again.
Only 3 queries are required (AI, player, draw), but the logic for the queries is more extensive, up to 81 logic elements are required.



It is evaluated horizontally, vertically and diagonally whether the AI occupies all 3 fields.



In the event of a draw, the query is slightly different, but basically the same:
You are asked whether all 9 fields are occupied, regardless of whether they are occupied by the AI or the player.

5) How do I tell the AI that it is its turn?

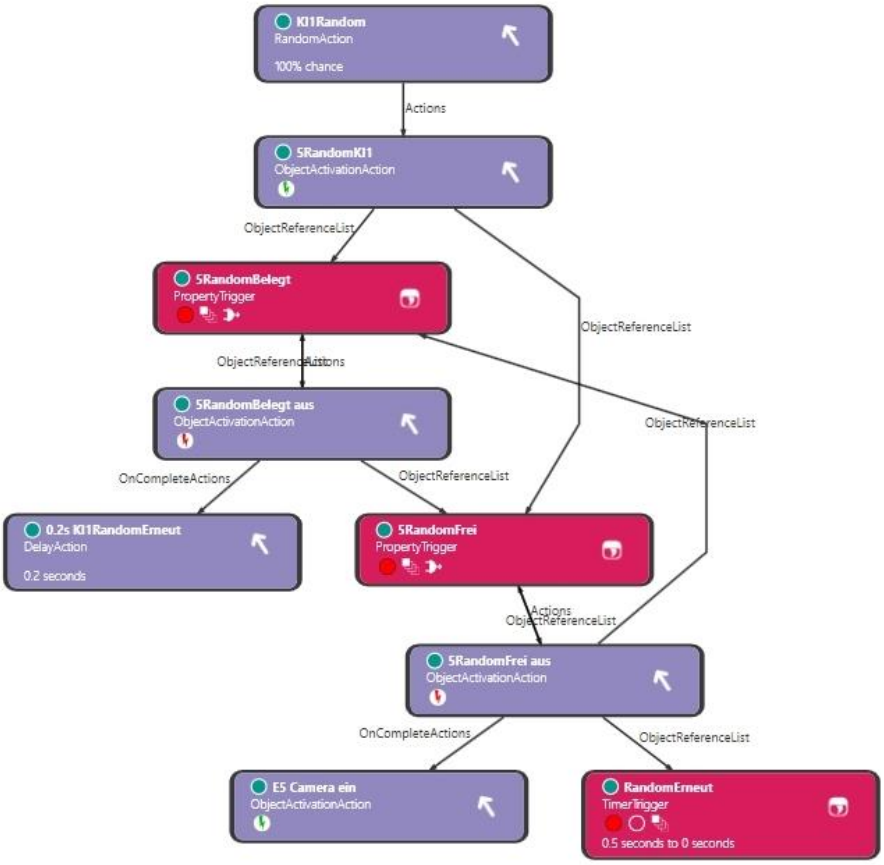
So that the AI “knows” when it is its turn, I could simply activate the AI every time the player occupies a square.
Unfortunately, it's not that simple, because there are situations in which the AI shouldn't make a move, e.g. when the player has 3 in a row, i.e. wins.
Therefore, after the player move, it must first be evaluated whether the game has been won or whether all squares are already occupied (= draw).
Therefore, a player move activates the playing field evaluation and a proximity action (called AI move), which triggers the AI move when leaving (OnExit) the associated area “AI move”.
By the way:
Why so complicated, you could start the AI move after completing the board evaluation?
That doesn't work because the board evaluation is also started after every AI move. So another AI move would be triggered all the time and the player would never get a turn.
Ok, if you mean that you could do 2 field evaluations, one for the player and one for the AI, you are right, but 183 objects are required for the evaluation.
As already mentioned, start the board evaluation and proximity action, as soon as the board evaluation has neither shown “Player has won” nor “Draw”, it triggers the removal of the AI move area.

Removing the area works in the same way as placing the tiles; the area is simply moved to a different location.
Because the proximity action reacts to leaving (On Exit), moving the area to a different location is the same as leaving the area.
If the evaluation shows that the player has won or there are no free spaces, the ProximityAction (AI move) is deactivated.
Deactivation means that no AI move is triggered when leaving the associated area.
Incidentally, the area is so large that the player will not leave it, even if he flies with a fighter jet at full speed.
Initially I had planned the area so that it would be left after 0.5 seconds of flight, but this caused further problems, e.g. if you fly with a helicopter and are not far away from the playing field, or fly out of the area with a fighter jet at full speed before the evaluation is finished.

6) How do I “program” an (albeit simple) AI with actions, areas, triggers and objects alone?

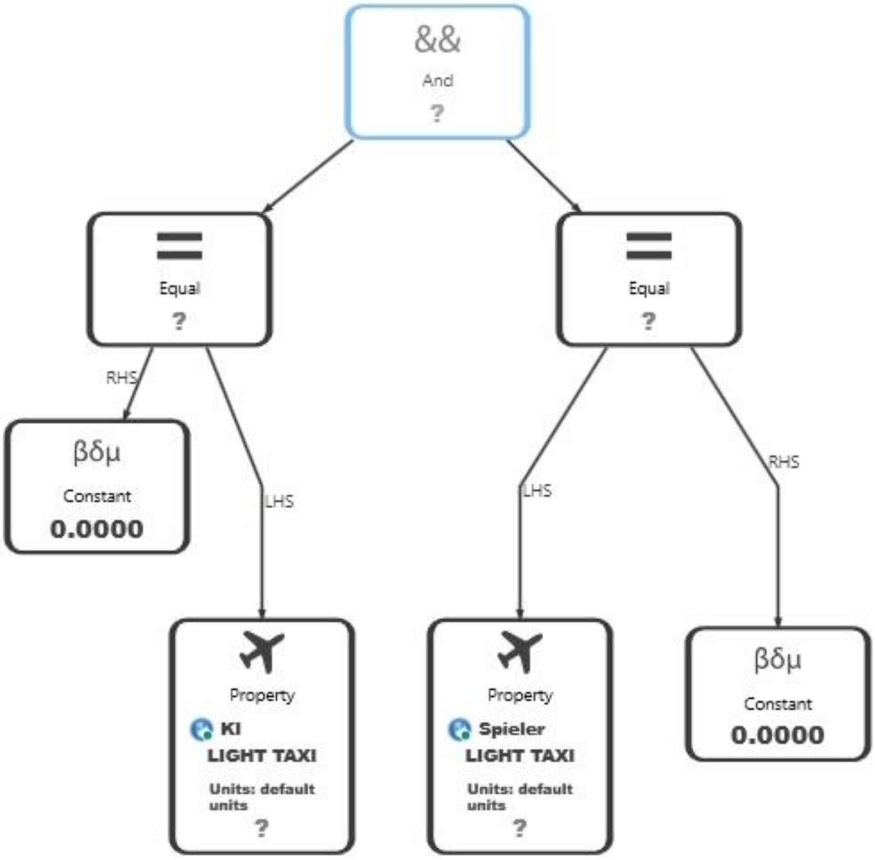
As soon as it is the AI's turn, a RandomAction is triggered.
This action randomly selects one of the 9 fields and activates two property triggers (free/occupied).

If “Free”, the AI occupies the field; if “Occupied”, the RandomAction is triggered again.
For some unknown reason, this only worked in 95% of cases, so I triggered the RandomAction again with a 0.5s delay. If the field is free, this delayed action is deactivated immediately.

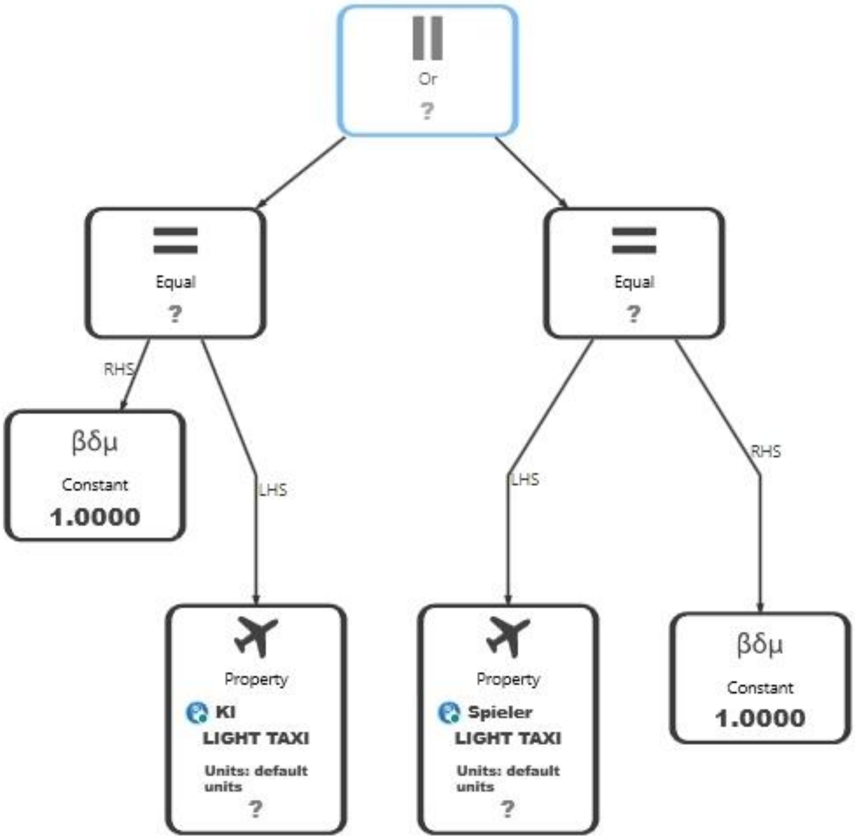


The logic of “Free” is quite simple:

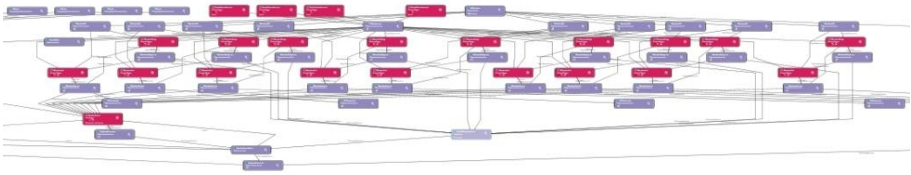
It is queried whether Light Taxi is switched off for AI and player airplane (Constant=0).



The logic of “Occupied” is corresponding:
It is queried whether Light Taxi is switched on for the AI or player aircraft (Constant=1).



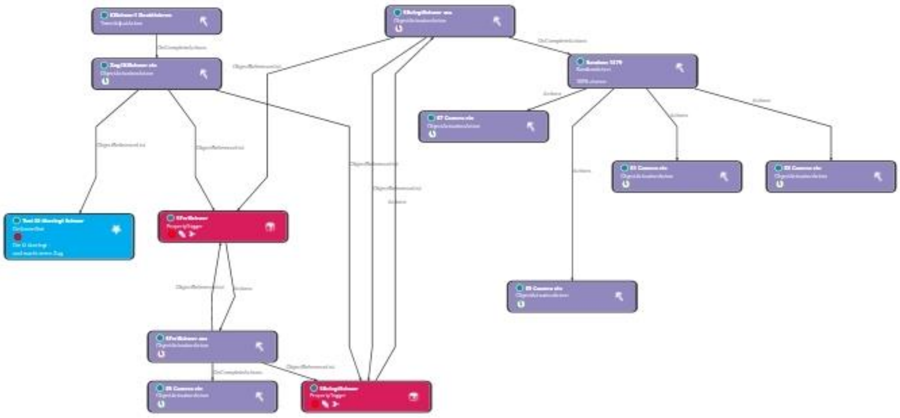
Individually this looks clear, but if you do this with all 9 fields and link them accordingly, 65 objects are required:



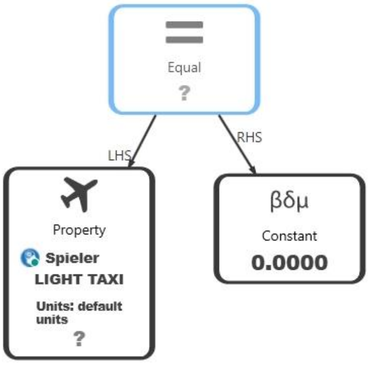
AI-Light chooses a free field at random, how do I make the AI intelligent?

7) How do I program an intelligent AI? There are many different game situations that the AI has to recognize / query in order not to lose.
The intelligent AI was a big challenge.
Especially move 2 was difficult to realize.
Because it is precisely move 2 that decides whether the AI can win.

But first things first:
1st move AI: Here it is simply asked whether field 5 (the middle one) is occupied, if no, the field is occupied by the AI, if occupied, field 1, 3, 7, or 9 is randomly occupied.



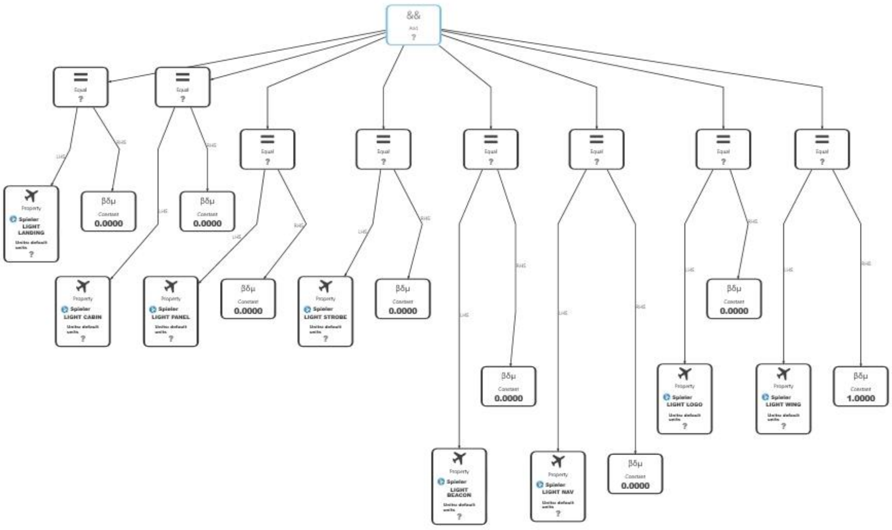
The associated logic is correspondingly simple:



Turn 2: Now it gets complicated. The AI does not know whether the player or the AI has started. A query as to whether field 5 belongs to the AI is not meaningful, because the player may have started with any other field and the AI only responded with field 5 on the first move.

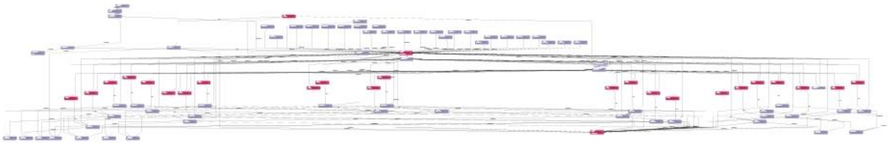
- If field 5 is occupied by the player, the further moves are simple: The AI simply asks whether the player has occupied 2 fields in a row and the 3rd field is free. So this 3rd square is occupied.
- If field 5 is occupied by the AI, it is possible that the player has already made one or two moves. For this reason, the system first checks whether only 1 field (on fields 1,2,3,4,6,7,8,9) is occupied.

This is one of 8 logics, here you are asked whether Wing (field 9) is occupied and all other 7 are free:

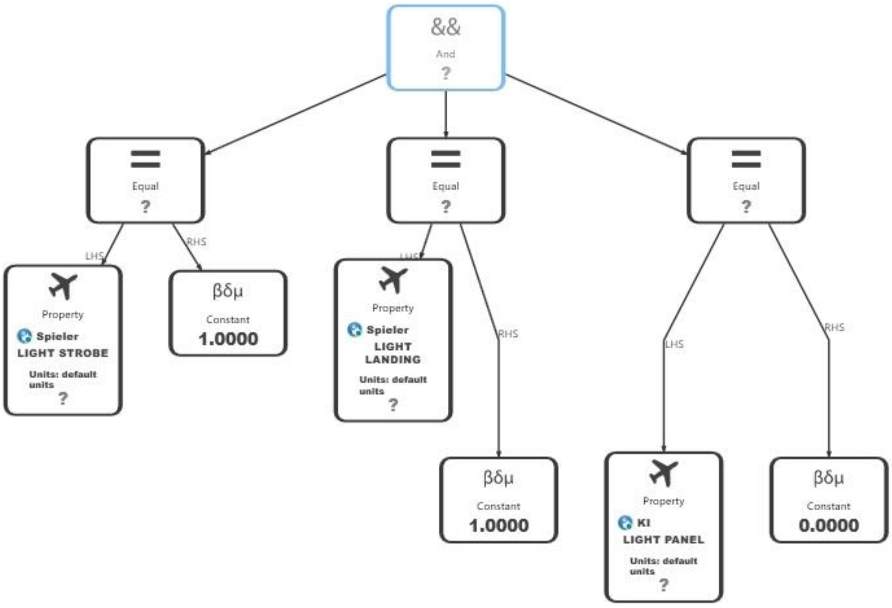


If one of the logics strikes, one of 2 answer fields is selected at random, in the above example field 3 or 7.

If no logic strikes, all possible combinations of 2 fields (on fields 1,2,3,4,6,7,8,9) must be queried.
Therefore, a query is first run after 2 squares occupied by the player in a row (horizontal, vertical and diagonal).
From now on, this query is run for every further move, by the way, there are 56 objects:

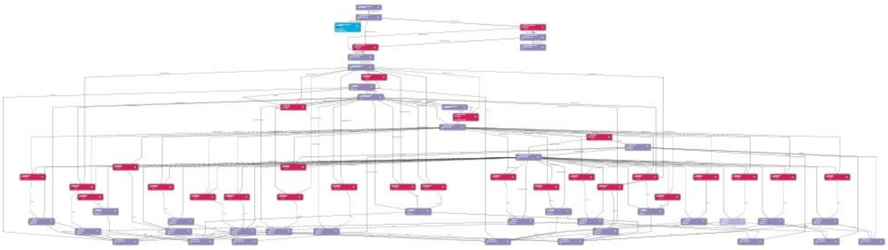


The logic looks like this. Here, one of 3 variants is queried in the upper horizontal row:

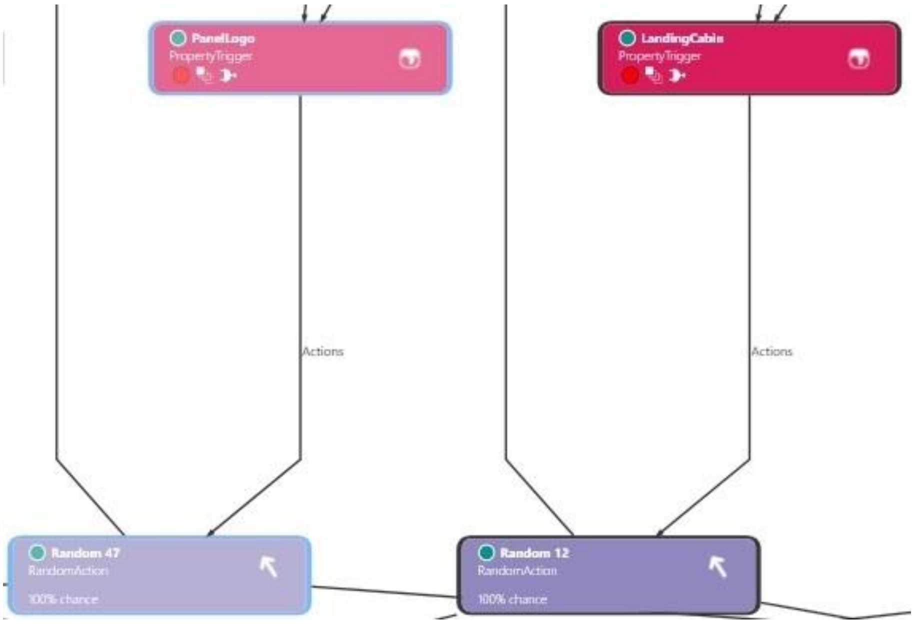


Wenn eine der Logiken anschlägt, wird das 3. Feld belegt.

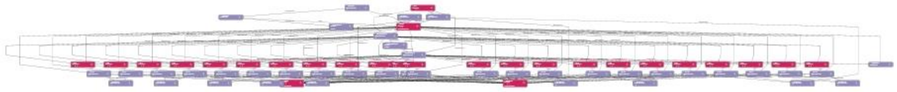
Wenn keine Logik anschlägt, müssen die übrigen möglichen Kombinationen abgefragt werden, z.b. Feld 1 und 8 belegt, in diesem Fall wird per Random Feld 4, oder 7 belegt.
Diese Abfrage benötigt weitere 56 Objekte:



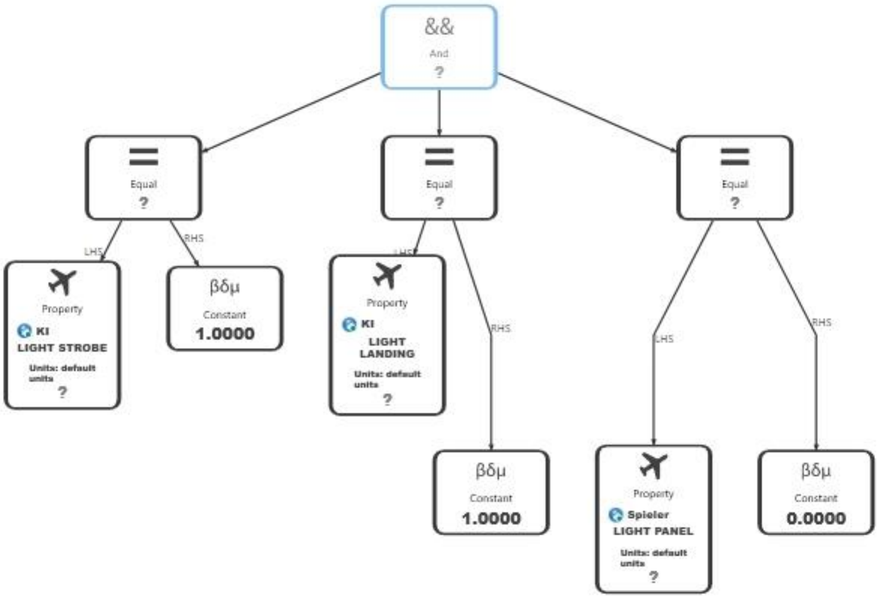
Here is a detailed view:



3. Zug: Ab hier wird es einfacher, denn die Züge 3 bis 5 laufen alle gleich ab:
Zuerst wird abgefragt, ob die KI 2 Felder in einer Reihe belegt und das 3. Feld frei ist.
54 Objekte wurden dafür benötigt:



The logic looks like this:



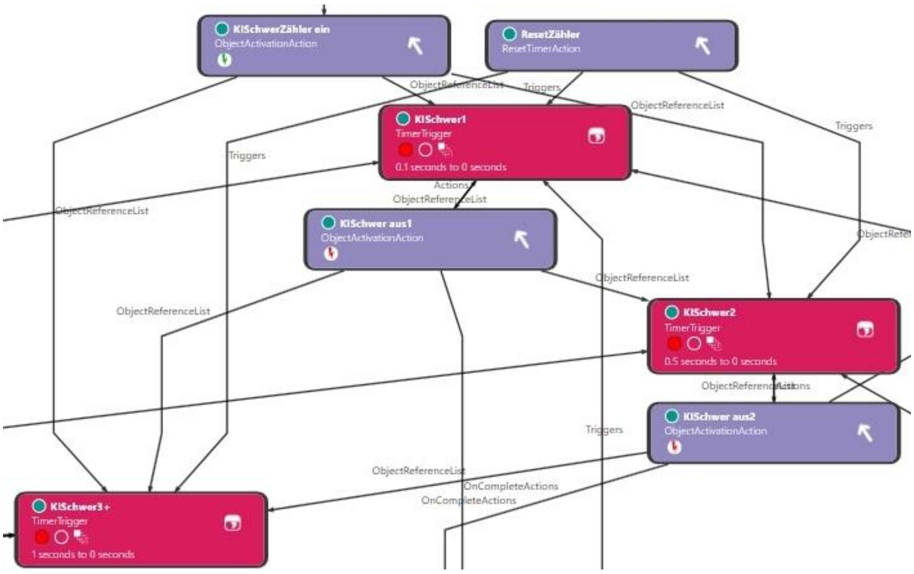
If one of the logics is successful, the 3rd field is occupied and the AI wins.

If none of the logics are triggered, the query already used in turn 2 is started.
If 2 squares are occupied by the player, the AI takes the 3rd square to prevent the player from winning.

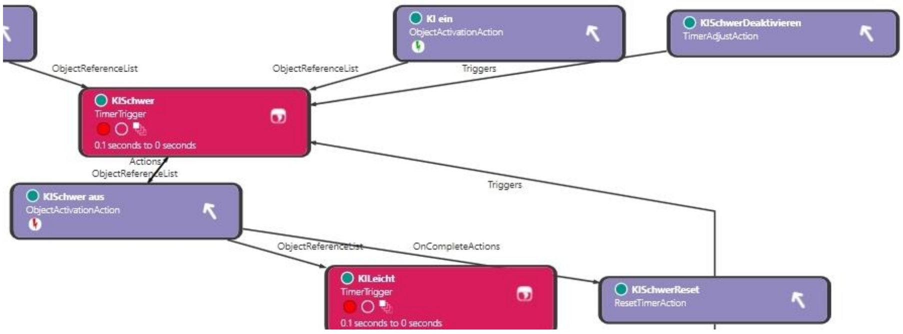
If none of the logic works, the random action of the “simple AI” is started, i.e. any free square is occupied, because a victory is no longer possible.

There was only 1 problem left to solve: How do I count the moves?
The AI must know whether it is making move 1, 2 or 3 so that the correct logic sequence is queried.
You could take 3 CounterTriggers, one counts to 1, one to 2, and the third to 3. Before each move you give all 3 counters a “tick” (the counters increase by 1 each time), and fire when the appropriate value is reached.
The problem with the CounterTriggers is that you cannot set them back to zero. That's why they can't be used, because they have to be reset after every game round.

After a lot of tinkering, I came up with TimeTrigger as an alternative.
3 timers (one with 0.1s, one with 0.5s and one with 1s) are started before the AI move.
- The 0.1s timer starts first and immediately stops all 3 timers. Then 1000s are added to the 0.1s trigger so that it does not fire the next time. Move 1 is started.
- Before the next move, the 3 timers are started again, this time the 0.5s timer fires, etc.



8) How do I differentiate between simple and intelligent AI after starting the game?
Using the same method as for counting the moves (above), I have created 2 TimeTrigger triggers.
Both timers are set to 0.1s. At the start of the game, a choice is made between easy and hard, which results in 1000s being added to the other timer.
Each time it is the AI's turn, both timers are started. Logically, the timer starts at 0.1s, switches off both timers and resets itself so that it is ready again at 0.1s for the next query. It then fires the appropriate AI (simple or intelligent).



I realize that the above descriptions are not complete.
The evaluation of who won, how the score display works, integration of the goal, etc. are still missing.
But with these descriptions you understand the principle behind it and if you want to know more, you can now open this mission in SimDirector and analyze it.

I hope you enjoyed this flight, if so please give feedback to p3d@andi20.ch . Also send error messages (spelling mistakes, wrong information, etc.) to p3d@andi20.ch, I appreciate any feedback.