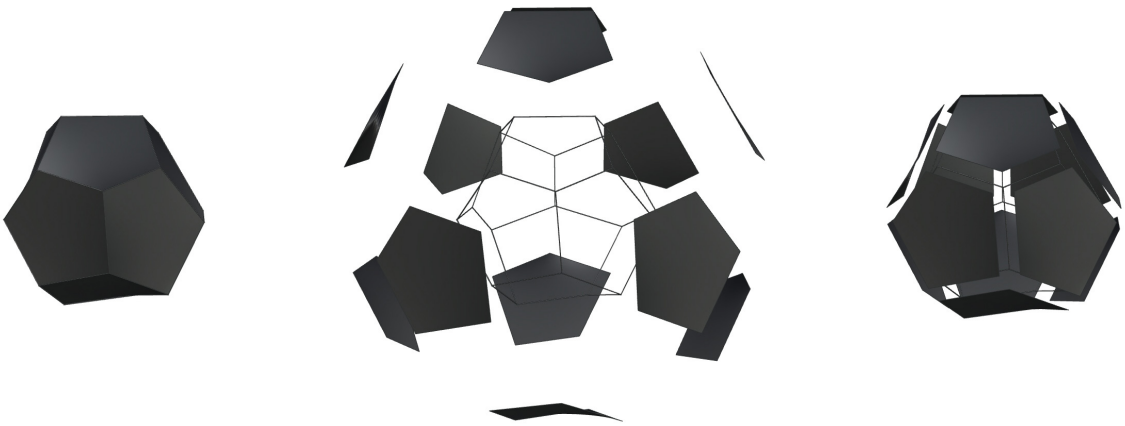


The Dodecahedron Puzzle Solver

Benjamin Felbrich

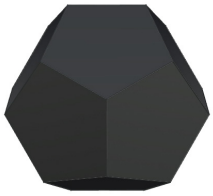
Dresden University of Technology
School of Architecture
Chair of Knowledge Architecture
Jun. Prof. Joerg Rainer Noennig

The Dodecahedron Puzzle Solver (DPS)

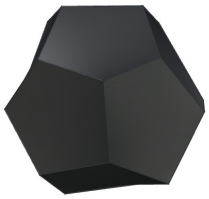
In his book „Mentopolis“ Marvin Minsky, co-founder of Massachusetts Institute of Technology‘s AI laboratory describes the human mind as an intelligent entity, composed of different parts of conscisnous which he called agencies. Every agency is capable of handling a relatively simple task by again subdividing it into sereval subtasks and distributing them over sereval agents. A fundamental thesis of his is, that there is no central unit controlling and organzing the distribution of tasks but that conciousness and intelligence are more of a result of the behaviour of interconnected agencies. Furthermore he postulates that it is possible to create artificial intelligence in computers by using this structure of seemingly non organized, relatively indepedent agencies.

The purpose of this work is to give an example of how to subdivide a relatviley complex task into simple subtasks in computation. Using Grasshopper and Rhinoceros 5 I made a small three dimensional puzzle from a distorted dodecahedron, dissembled its pieces and let a little script try to reassemble them back together. I call this algorithm the Dodecahedron Puzzle Solver (DPS). Furthermore I want to show, that the solver runs more efficient, when it is able to utilize some kind of memory. I call this algorithm DPSm (dodecahedron puzzle solver with memory).

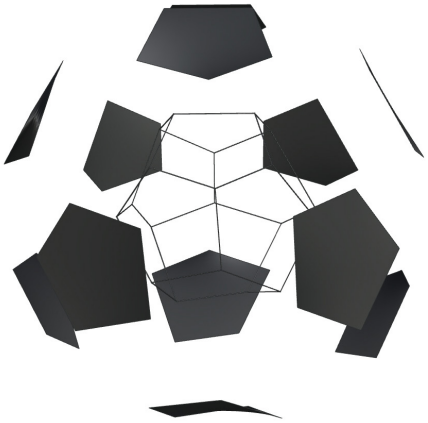
The dodecahderon is distorted in a way, that every edge has a unique length. Thus each and every piece of the dodecahderon is unique and only fits into exactly one position. Thus there is only one solution to the puzzle.



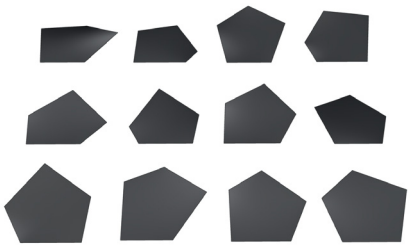
1 regular dodecahedron



2 distorted dodecahedron



3 dissembled pieces

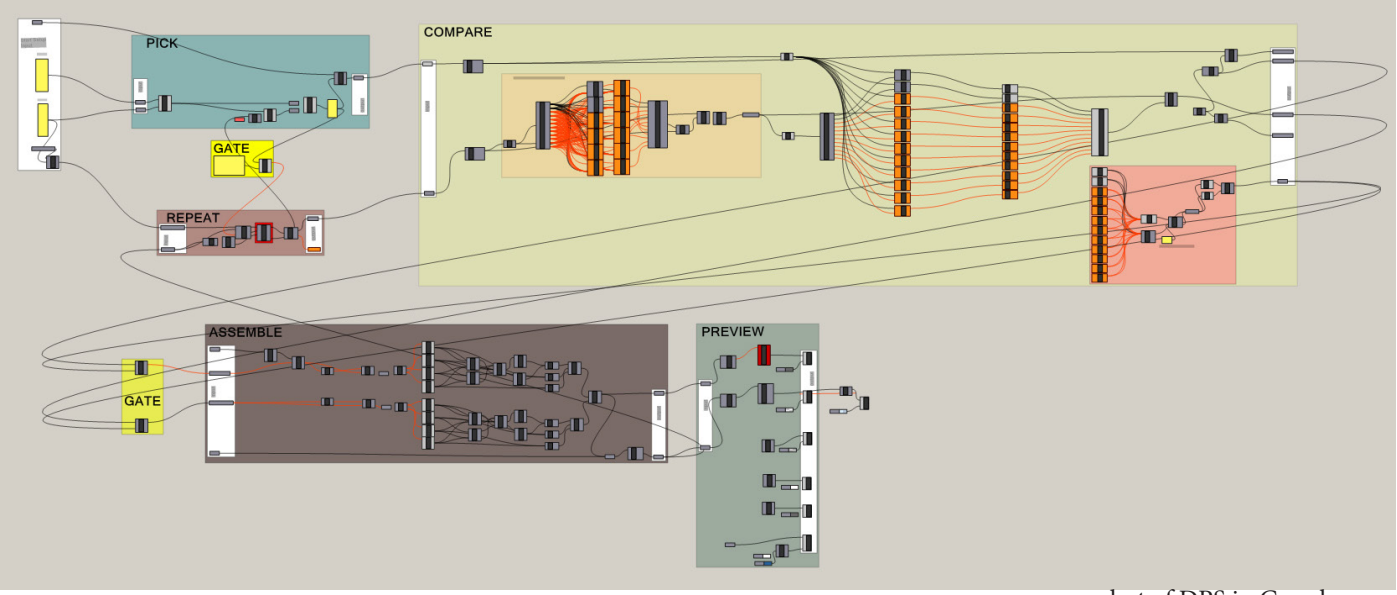


4 shuffled and rearranged

5 reassembly

Reassembling the pieces step by step into its starting position is a relatively complex task. In the following I want to give some general information on how the scripts work and then go into detail by explaining the methods of operation of the single agencies. Finally i want to give a quick overview on the comparison of DPS and DPSm.

general info: The script recognizes each piece as the list of its edge curves and their lengthes. Thus each piece consists of five numbers.
The length of each edge curve is unique and therefor stands as their ID in the script.
During its work, the script has no information of any kind about the final solution, it‘s work is solely based on try and error.
P stands for the „pick“ meaning the piece that is chosen to assemble.
E stands for „existing“ meaning the collection of pieces that are already assembled (the partial solution).



screenshot of DPS in Grasshopper

agencies of DPS:

PICK

input: collection of ten shuffled, unorganized, available puzzle pieces
output: one pick from the collection of available pieces (P)
procedure: randomly pick one piece from the set

COMPARE

input: P
E
output: boolean statement if P generally fits in E at any position
index of the edge curves in P that fit in as neighbor curves E
index of the curves in E that fit in as neighbor curves of P
procedure: retrieve every edge curve of P
retrieve every edge curve of every piece in E
compare the edge curves in E by their length and retrieve those which only appear once (and are therefor „naked“ edges)
compare the edge curves in P with the naked edge curves in E by their length
if at least two edge curves in P have an equal length to those in E, the piece fits

ASSEMBLE

input: P
E
index of fitting edge curves in P
index of fitting edge curves in E

output: new E with the new piece assembled (newE)

ASSEMBLE is only executed, if COMPARE returns the value „true“ on the fitting request

procedure: sort the fitting edge curves in P by their length
sort the fitting edge curves in E by their length
create a unique definite plane Pl1 based on the starting and end points of the edge curves in P
create a unique definite plane Pl2 based on the starting and end points of the edge curves in E
rearrange P by using Pl1 as the base and Pl2 as the target position in space

PREVIEW

input: P
E
relevant edges

ouput: Preview

procedure: show everything hued nicely

REPEAT

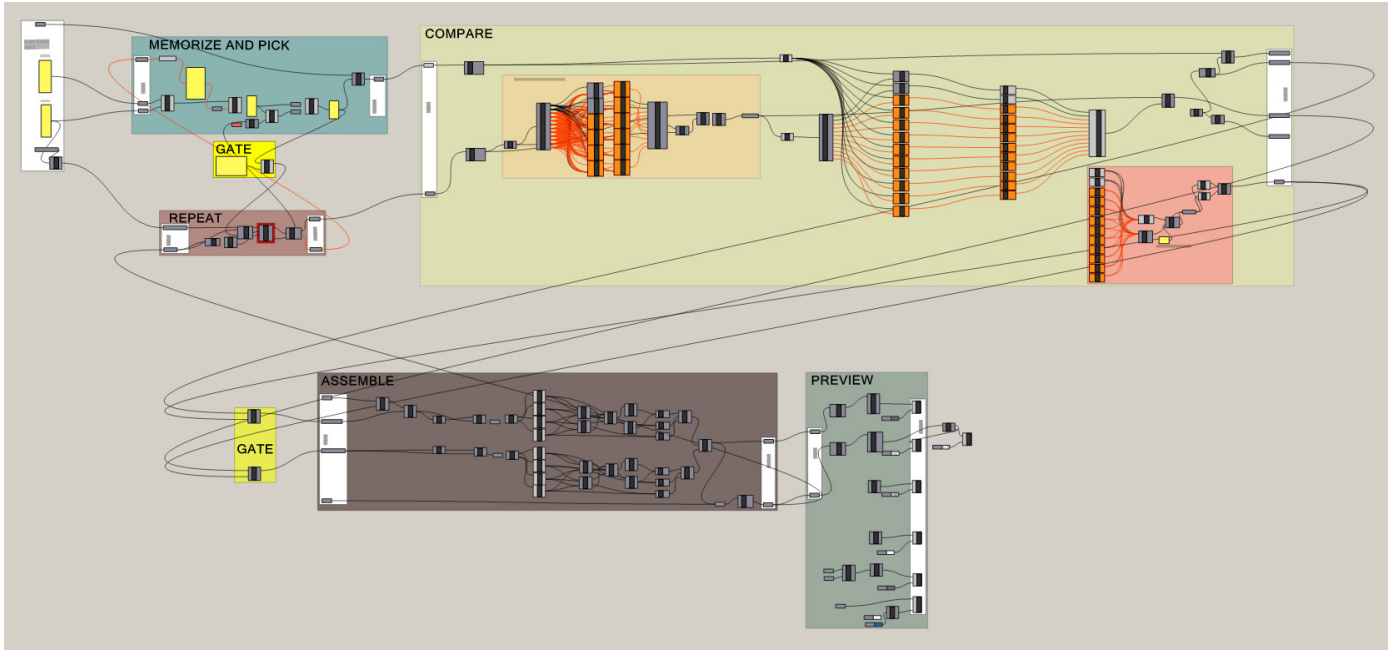
input: newE

ouput: E

procedure: take the newly assembled pieces and put them back to the start of the script to repeat the whole thing
stop repeating when the puzzle is complete

DPSm

The dodecahedron puzzle solver that utilizes memory works exactly like it‘s simpler equivalent with small differences in **PICK** and **REPEAT**. Whenever a fitting piece is found and assembled, DPSm eliminates this piece from the list of possible piece to pick. In other words it „memorizes“ the previous successful picks.



screenshot of DPSm in Grasshopper

result:

By testing DPS and DPSm numerous times and variing the engine for the random pick it was possible to compare their efficiency. By applying the small changes and including a memory, DPSm returns the correct solution with approximately three times lesser picks than DPS as the following table shows.

Pass	number of picks to solution	
	DPS	DPSm
1	27	28
2	89	14
3	45	18
4	67	15
5	56	17
6	34	14
7	23	12
8	17	16
9	26	16
10	25	15
11	78	19
12	34	13
13	24	13
14	50	32
15	39	21
16	100	20
Ø	45,9	17,7

The video „DPSvsDPSm_Benjamin_Felbrich_130810.wma“ shows an example of how DPS and DPSm work. The relevant files for Rhino and Grasshopper are also enclosed.

appendix:

DPSvsDPSm_Benjamin_Felbrich_130810.wma
puzzle_solver_130808.3dm
DPS_130812.gh
DPSm_130812.gh

sources:

„Mentopolis“, by Marvin Minsky; Klett-Cotta 1985

software:

Rhinoceros version 5 SR 3
Grasshopper version Build 0.9.0056
Hoopsnake version 0.6.4 by Yannis Chatzikonstantinou