by Markus Krugel

Progress report

of the Project „Creature Generator“

Table of Contents

[Goals and purpose: 2](#_Toc2203035)

[Algorithms and Concepts 2](#_Toc2203036)

[Poisson disc sampling: 2](#_Toc2203037)

[Superformula: 3](#_Toc2203038)

[Divago system: 3](#_Toc2203039)

[variational auto includer: 4](#_Toc2203040)

[Multi-resolution surfaces 5](#_Toc2203041)

[Lindenmayer system: 5](#_Toc2203042)

[metaballs: 5](#_Toc2203043)

[rigblocks 5](#_Toc2203044)

[Potential solutions 6](#_Toc2203045)

[My approach 6](#_Toc2203046)

[Project Plan 6](#_Toc2203047)

[Gantt chart 6](#_Toc2203048)

[References 6](#_Toc2203049)

# Goals and purpose:

The minimum goal of this project is that the user can set the settings and then a monster, with basic body parts merged into a single mesh, will be generated. Furthermore, the user will have the ability to export the monster model so that they can use it outside of this project, for example in a 3D modelling software. When I reached the basic goal and still have time remaining I will consider implementing other features like additional body parts or further control for the user on the creature generation inside the editor.

Procedural generation takes a big part of the video game industry as seen in their big role in genres like roguelike or endless runners. While the generation of worlds and levels are the main purpose of procedural generation in games, the generation of creatures are rarely seen. Only a few games like No Man’s Sky tackle this issue. My intention behind this project is to explore possibilities in the creation of creatures as I can see many benefits it can have for the video game industry.

One advantage is that artists can use these generated models as a start to create unique monsters or even inspire them to create ones on their own. Moreover, it can help to reduce the time and cost behind developing the creatures as they can be generated instead of being modelled from the ground. The procedural generation of creatures can be also interesting for the players as it can motivate them to explore the world as they can see more of these unique creatures. One example for the last advantage can be seen in the mentioned game No Man’s Sky as some players like to spend more time exploring due to these vastly different creatures.

# Algorithms and Concepts

## Poisson disc sampling:

Poisson disc sampling is an algorithm to generate multiple objects which have a minimal distance between each other, so that they do not collide.

Instead of just spawning the object randomly and then going through every existing object to check if they are colliding, we implement a grid system. The objects now have the length of a cell’s diagonal as their radius so that it is guaranteed that they fill the cell despite their position inside the cell.

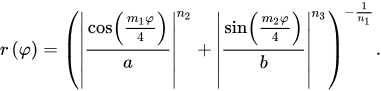
Furthermore, because of their radius size we only have to consider a 5x5 area with the cell as its center, which helps vastly to improve the efficiency [2].

This easily implemented algorithm can be used for the project to place the body parts on the body without fearing that they collide with each other.

On the other side this algorithm does not help with the problems of creating the body parts and the combination of these into a single mesh. Moreover, we have to constraint the places where the body parts can spawn as this can result into strange creatures, for example the feet could be spawned on the upper part of the body where they would be useless.

## Superformula:

The superformula is an algorithm, proposed by Johan Gielis around 2000. This formula is a combination of the equations for spheres, eclipses and the superellipse and can be used to create different shapes.



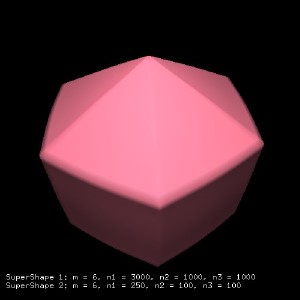
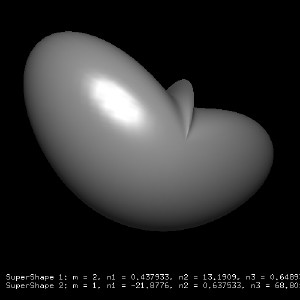
The values of the parameters a, b, m1, m2, n1 and n2 can be changed to achieve different results. As this equation can only have two dimensional results, we have to multiply two different superformulas as following:

https://lh6.googleusercontent.com/E3NqWr5N5_BkHinwp3w2GyZdWQoppK_JrTIFerCeEskcosByyP_KaLJrR6OZyxRrVSg8EZK3VDSlqWOYXoCMIW8pPt3uW9K8AiAnC9Las8J-8l1dirZF34LW8oYBKwRWClh5SnLQ

https://lh5.googleusercontent.com/7OuOEsbnZYKZORiA_0ixXMmsqz12HXkkAuW-JAjNKMwRAYg63lUKtU_vhffcB_0JoT70V-9xKeUBEvPuwt3DuG7WI79yqv0S62LTpM2RzvsAvjxtkZrv4ovUAVZa78hI662b1-aV

https://lh4.googleusercontent.com/2o1F4hsFcBpSPUL6Uf-82np71QJioRwLA5YFsrA-rFdIQ_auA_ZpDIdr8hoS98wbKgZNEKrwuiv5xhg80Mlzb9cxAssgb0hpbIn9nL8cwEbqKxdA-Fxf9Dq9jAHBL6lVc7tM6dJJ

The https://lh6.googleusercontent.com/bqkLQ0L0-MMBGufaJ1TZnIaHhbDSPmwuWMnT_0bS5j5H0nIzwzgFdmP8wxDc5njYOtQWyxocTq_TfA-klfOF3rRs3AtTEwHZ6OVLzFa-Xl8Ear6yXw1jyU-fGqaaR3UvR6Q9EXvq can only have values between -*π*/2 and *π*/2 while *θ* stays between -*π* and *π* [6]*.*

 [10]  [11]

*Examples of Supershapes in 3D.*

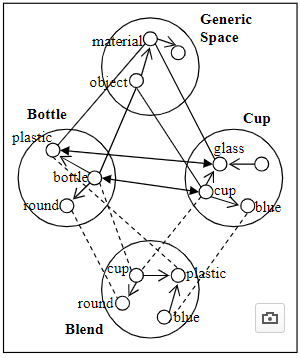
This formula could generate different shapes to be used as the body or for the body parts. Also, it is easy to use because you only have to change the values to have different shapes.

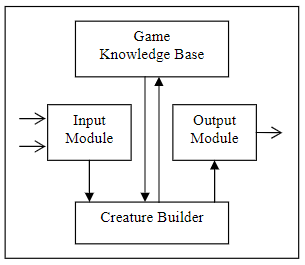
On the contrary it could generate shapes which are not usable to use, and it could be hard to find the usable shapes. Furthermore, there is unfortunately an active patent on this formula so that I can not use it for this project.

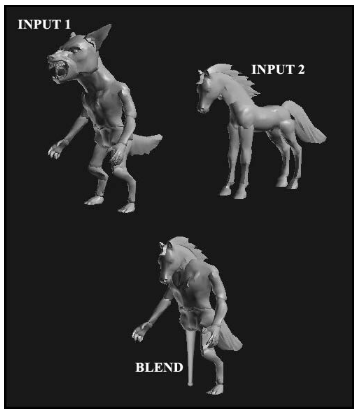
## Conceptual blending and Divago system:

The goal of the Divago system is to create an unique object by blending two different objects using knowledge gained by previous blends.

* blends two models into an unique new one

 [11]

[12]

[13]

* mapping phase: create mapping between components of each input
* projection phase: make blend projections
* elaboration phase: enhance/ refines the blends based on concepts from game knowledge base
* validation phase: checks if does not violate any rules

One disadvantage of this approach is that you need a few models beforehand as without the required inputs. As many learning algorithms it can take many data example before getting good results. Furthermore, the variations in the resulted blend can heavily be influenced by the differences between the inputs, which means that the data examples should have many differences to have varying results.

On the other hand once this system is trained the results could be very promising as it learns to understand how the creature should be generated.

## variational auto includer:

* probability model perspective
  + d
* neural net perspective
  + encoder
    - puts input into an output with lower dimensions
  + decoder
    - decodes output of the encoder
  + loss function
    - calculates the error between original and reconstructed
* problem: needs data set to be trained

## Multi-resolution surfaces

* divide the mesh into subdivisíons
* add details to the subdivisions

## Lindenmayer system:

The Lindenmayer system or also known as L-System is a string rewriting algorithm developed by Aristid Lindenmayer in 1968.

You start with a base string and with each recursion you replace this string based on the rules you set beforehand.

Following we have an example of this algorithm. We start with the string “E” and the rules are “E = ER” and “R = E” which means that with every recursion we replace the all E’s in the string with “ER” and all R’s will become E’s.

n=0:         E   
            / \  
n=1:       E R

         /| \  
n=2:     E R E          
        /| |   | \  
n=3:   E R E   E R

    / | | | \   |\ \  
n=4: E R E E R  E R E

So, after four iterations our initial string of “E” have become the string “EREERERE”. At the end we will execute operations based on the letter. In this example “E” means “Extrude” and ”R” is “Rotate”.

For our cause a stochastic grammar would be more useful as it can have different results. As an example we can change the rules from the previous example by saying there is a 50% chance of “B = B” and the remaining 50% will result in “B = F”.

An advantage of this algorithm is that it is easy to understand and to implement. Additionally you can easily modify the rules if the results are not satisfying.

The biggest problem with this approach is to find a correct grammar to generate acceptable creatures.

## metaballs:

* balls merges when nearby
* simple shapes (box, circle, ellipse)
* could be used for the body or for really simple creatures

## Rigblocks

Rigblocks are models with components, like the width of the nostrils, which can be modified by the user.

The game „Spore“ uses this system inside their creature editor so that the player can deform and modify the body parts to their liking [3].The developers behind spore used sliders to change the components because they are intuitive for the player to use. Additionally, these models also contain animations, which are being played when the user deforms the model so that there is a fluent transition between the old and new state [8].

[14]

*On the left we see the model with its handles, the following images show possible deformations*

As seen in Spore this system is really helpful in creating quality body parts, which can be deformed and therefore have many variations. Furthermore, this could help to implement features similar to Spore’s creature editor where the player can adjust the creature inside the editor after the generation.

The problem for this system to this project is that the integration of these body parts with the body could be difficult because we have to find a solution to combine the different meshes into a single mesh without strange deformations of the body or the body parts.

# Potential solutions

# My approach

# Project Plan

## Gantt chart

**Timetable**:

* to next week: bit more research, decision on the generation approach
* 1 week for editor GUI
* 2 - 3 weeks for implementation of the algorithm
* 1 week for model export
* 1 - 2 weeks for testing, fixes and optimizations
* 1 week for finalizing the report
* 1 week buffer time

# References

## Text

1. Ribeiro, P., Pereira, F., Marques, B., Leitão, B. and Cardoso, A. (2019). A Model for Creativity in Creature Generation. In: *4th International Conference on Intelligent Games and Simulation {(GAME-ON}*. [online] London: EUROSIS, p.5. Available at: <https://www.researchgate.net/publication/221024396_A_Model_for_Creativity_in_Creature_Generation> [Accessed 27 Feb. 2019].
2. [Unity] Procedural Object Placement (E01: poisson disc sampling) (2019) YouTube video, added by Sebastian Lague [Online]. Available at <https://www.youtube.com/watch?v=7WcmyxyFO7o> [Accessed 27 February 2019].
3. Hecker, C. (2014). *My liner notes for spore*. [online] Chrishecker.com. Available at: <http://chrishecker.com/My_liner_notes_for_spore> [Accessed 27 Feb. 2019].
4. ‘Metaballs’ (2019) *Wikipedia.*Available at <https://en.wikipedia.org/wiki/Metaballs> (Accessed: 27 February 2019).
5. Wong, J. (2014). *Metaballs and Marching Squares*. [online] Jamie Wong. Available at: <http://jamie-wong.com/2014/08/19/metaballs-and-marching-squares/> [Accessed 27 Feb. 2019].
6. ‘Superformula (2018) *Wikipedia.*Available at <https://en.wikipedia.org/wiki/Superformula> (Accessed: 27 February 2019).
7. Velho, L., Perlin, K., Biermann, H. and Ying, L. (2002). *Algorithmic shape modeling with subdivision surfaces*. [ebook] New York: IMPA—Instituto de Matematica Pura e Aplicada, pp.865 - 875. Available at: <https://web.stanford.edu/~lexing/asmss.pdf> [Accessed 27 Feb. 2019].
8. Choy, L., Ingram, R., Quigley, O., Sharp, B. and Willmott, A. (2007). *Rigblocks: Player-deformable Objects*. [ebook] Electronic Arts. Available at: <http://www.cs.cmu.edu/~ajw/s2007/0248-Rigblocks.pdf> [Accessed 27 Feb. 2019].

* <http://www.makinggames.biz/feature/procedural-generation-characters,12342.html>
* <https://www.researchgate.net/publication/221363266_Segmentation_of_Building_Facades_Using_Procedural_Shape_Priors>
* <https://jaan.io/what-is-variational-autoencoder-vae-tutorial/>
* <https://www.researchgate.net/publication/316968105_Style_and_Type_in_a_Generic_Shape_Grammar_The_Case_of_Multipurpose_Chairs>

## Images

1. Bourke, P. (2002). *Example 1 of Supershape in 3D*. [image] Available at: <http://paulbourke.net/geometry/supershape/C_0009s.jpg> [Accessed 27 Feb. 2019].
2. Bourke, P. (2002). *Example 2 of Supershape in 3D*. [image] Available at: <http://paulbourke.net/geometry/supershape/C_0022s.jpg> [Accessed 27 Feb. 2019].
3. Cardoso, A. (2003). *Conceptual blending figure*. [image] Available at: <https://www.researchgate.net/publication/221024396_A_Model_for_Creativity_in_Creature_Generation> [Accessed 27 Feb. 2019].
4. Cardoso, A. (2003). *Divago system figure*. [image] Available at: <https://www.researchgate.net/publication/221024396_A_Model_for_Creativity_in_Creature_Generation> [Accessed 27 Feb. 2019].
5. Cardoso, A. (2003). *Outcome of the Divago system*. [image] Available at: <https://www.researchgate.net/publication/221024396_A_Model_for_Creativity_in_Creature_Generation> [Accessed 27 Feb. 2019].
6. Choy, L., Ingram, R., Quigley, O., Sharp, B. and Willmott, A. (2007). Example of a rigblock. [image] Available at: <http://www.cs.cmu.edu/~ajw/s2007/0248-Rigblocks.pdf> [Accessed 28 Feb. 2019].