IT UNIVERSITY OF COPENHAGEN

Master Thesis

Interactive Evolutionary Computation in Architecture: Interactive Braid Evolution

Author: Patrikk Sørensen
Supervisor: Sebastian Risi

IT University of Copenhagen
Copenhagen, Denmark

A thesis submitted in fulfillment of the requirements for the degree of Master of Science.

December 2016

Abstract

Intereactive evolutionary computation has been used in various domains where personal preferences can help guide the evolution and search the problem space of a given domain. These projects have been especially useful for creating art, pictures, 3d shapes in which it is hard to create a general fitness function that can find a global optimum of something that every person evaluates as the best solution. In this project, interactive evolutionary computation is explored inside the field of architetucture and in the context of the ressearch project Flora Robotica. Here a certain category of artifacts defined as braid structures is evolved by users through CPPNs and the NEAT algorithm, creating innovative braid structures that can be used with external processes from the Flora Robotica project. The results compares the user evolved braid structures with the handmade braid structures from the CITA studio, and then how different user strategies and interfaces can be integrated in the application loop in order to enhance the effeciency of interactive evolution, and how augmented reaility holds possibilities and challenges when designing an application with interactive braid evolution, in order to reflect on the future role of interactive evolutionary computation.

Keywords: Machine Learning, Interactive braid evolution, Augmented Reality, Architecture, Braiding patterns, 3d modeling, human-computer interaction,

Acknowledgments

The project would like to give a warm thanks for everyone at CITA studio for inviting me into their studio with a wonderful and creative atmosphere, and very lovely coffee. Thanks to the people who have been helping me get into the field of architecture, especially a big thanks to David and Petraz for always having time to discuss my application. Thanks to Phil for accepting and guiding my project into the CITA studio, and always be ready on the mail and respond to my practical boring questions. Thanks to Flora Robotica to allow me to work inside their very interesting domain of bio-hybrids. Last but not least, thanks to my wonderful better half Freja for all of her support during my hard times for finishing the thesis. It has been a great journey, that would have been hard to achieve if not for you.

Contents

List of Figures

List of Tables

Contents

1

Introduction and motivation

Technology for the last decade have provided new possibilities for living our everyday lives in the digital world, by introducing novel ways of interacting with computers, gadgets, robots, programs, the internet and other digital objects that has found their way into our home and is used in the world on a daily basis. These new possibilities allows for better communication, new ways of expression, for increasing the effectiveness on various work fields, for simulation and calculation, research and it is getting increasingly harder to think of an everyday life without some sort of influence from these technologies.

With new technologies emerging, along with the optimization of existing technologies and hardware, so too is the way we interact with the digital world changing by each year, and our use of those technologies is then changed, forgotten, introduced, created or removed, and while the speed of which humans can adapt new technologies and devices into their everyday life can vary from society, personality, and beliefs, it is interesting to see how so powerful technologies can fundamentally change our everyday life in a relative short period of a couple of years. On one side it is interesting to observe how these technologies can create innovative new ways of living, and on the other side it is getting increasingly interesting to see how we use and adapt these technologies to their own needs.

One of such technologies that seems to be currently relevant, is found in the robotics and machine learning field. While machine learning is certainly being embraced to either solve or optimize a long list of issues that is faced

by our societies, so too is the relevance for exploring how we can interact and adapt machine learning for our own needs. One such technique is defined as interactive evolutionary algorithms, in which humans can guide the algorithm to solve specific problems by evaluating the results produced by the algorithm. This algorithm can be used to create subjective solutions to problems that does not necessarily have a general solutions to it, but rather multiple solutions that is depending on the users preferences and personality. By exploring interactive evolutionary computation, we can get a deeper understanding on how one type of machine learning algorithm can be integrated and used in our everyday lives.

The main motivation for this project is then the question on how we can extend our possibilities to interact with the machine learning algorithm interactive evolutionary computation in order to create meaningful content that can be used in various and complex domains. This motivation is furthermore inspired by the introduction of the technologies of virtual reality and augmented reality, in which new possibilities with interacting with the digital world is once again expanded in novel and interesting ways and.

The field of artificial life and architecture provides an excellent testbed for exploring how interactive evolutionary computation can be further extended in creating novel ways of human-computer interaction with different domains, in which the research project Flora Robotica explores how plants, humans and robotics can live side by side in a symbiotic relationship. Here, artificial life is explored in order to find new possibilities of extending natural life as we traditionally know it by creating hybrids of mixed societies that are built from natural and robotic components. This projects serves as the main collaborator of this project, for exploring how ordinary users can create architectural artifacts that plants and robotic entities can inhabit.

1.1 Problem statement / hypothesis

The Interactive Braid Evolution project seeks to apply Interactive Evolutionary Computation in the context of architecture and artificial life, by collaborating with the research department CITA (Centre for Information Technology and Architecture) at the Royal Architecture School of Copenhagen, which is one of the universities working on Flora Robotica. The project builds on top of the concept *social garden* in which human/plant symbiotic interactions is promoted to develop cultural practices specific to the development of bio-hybrid structures. Inside the social garden, humans can then create structures in which plants and robotics can live side by side in and create a symbiotic relationship.

The project seeks to to be interdisciplinary and explore interactive evolutionary computation from two sides. On one side, it explores the possibilities for interactive evolutionary computation to create meaningful structures that can be inhabited by plants and robotics, and on the other side how interactive evolutionary computation can be influenced by different interfaces and input devices like augmented reaility in order to improve the effeciency of the algorithm. The structural possibilities is investigated by utilizing compositional pattern producing networks in combination with the evolutionary algorithm *Neuroevolution of Augmenting toplogies*, while the interfaces and input devices used in the interactive evolutionary computation algorithm is explored by comparing augmented reality devices with everyday input devices.

The hypothesis can be narrowed down to a project that: explores how humans can create braided structures through different interfaces and input devices by using interactive evolutionary computation and compositional pattern producing networks, and how can those braided structures contribute to a bio-hybrid system similar to the *social garden* described by the research project Flora Robotica.

2
Background

This section gives an overview of the projects background in the fields of interaction design and evolutionary algorithms applied in the project, covering the core principles, techniques and algorithms, along with a description of the domain the project takes place in. More specifically it covers the evaluation technique Interactive Evolutionary Computation, Flora Robotica, Neuroevolution, and the evolutionary algorithm Neuroevolution of augmenting toplogies. Each of those topics will be accompanied by example applications which will be highlighted from an algorithmic and design perspective that illustrates various ways of implementing interactive evolutionary computation.

Lastly, different architectural approaches for creating the braided structures that is used in Flora Robotica will be described.

2.1 Interaction design principles and terminology

The core terminology of interaction design is based on Normans book "The Design of Everyday Things"[19] and the more comprehensive book "Interaction design beyond human-computer interaction" [20]. This literature provides a useful terminology when dealing with interaction design, in particular for defining the terms: *mapping, feedback, signifiers, affordances* and *constraints*. Similarly, they provide models to analyze and describe user in-

teraction with systems, and principles for creating a good conceptual model of the application.

It should be noted, that while this literature also provides a great framework on the psychology and cultural dimensions on interaction design, especially in low-level cognitive psychology describing learning curves, memory, interpretation and recognizing meaning in the world, it is mainly used to the low-level human-computer interaction of the application, which is the reason for leaving some of those aspects out of this project, even though they can be of great importance.

2.1.1 The terminology of interaction design

Signifiers and *Affordances* is two related terms describing how possible actions can be communicated directly or indirectly to the user [11] [19]. While affordances describes the possible actions from an interactionist point of view [11], affordances can be confusing when dealing with virtual objects, giving rise to the term signifiers [19]. Affordances defines what actions are possible, both the intended and unintended actions, while signifiers in turn specify how those possibilities are discovered: signifiers are percitible signals of what can be done (e.g. icons). *Mapping* is a technical term describing the relationship between two objects, in this case between the operation performed, and what is it is mapped to have an effect on. A classical example is the light switch, in which switches to the matching light source typically resides in the same room.

Feedback is a familiar term from science, that describes how information is sent back to the user, notifying on the action being performed or what result has been accomplished. This power of feedback is illustrated by an exercise where one could try to talk without hearing the voice of themselves [19].

Constraints is another familiar term, that in this project describes how the system can restrict the user interactions that can take place at a given moment. This can be further divided into several categories of constraints:

physical, semantic, cultural, and logical constraints.

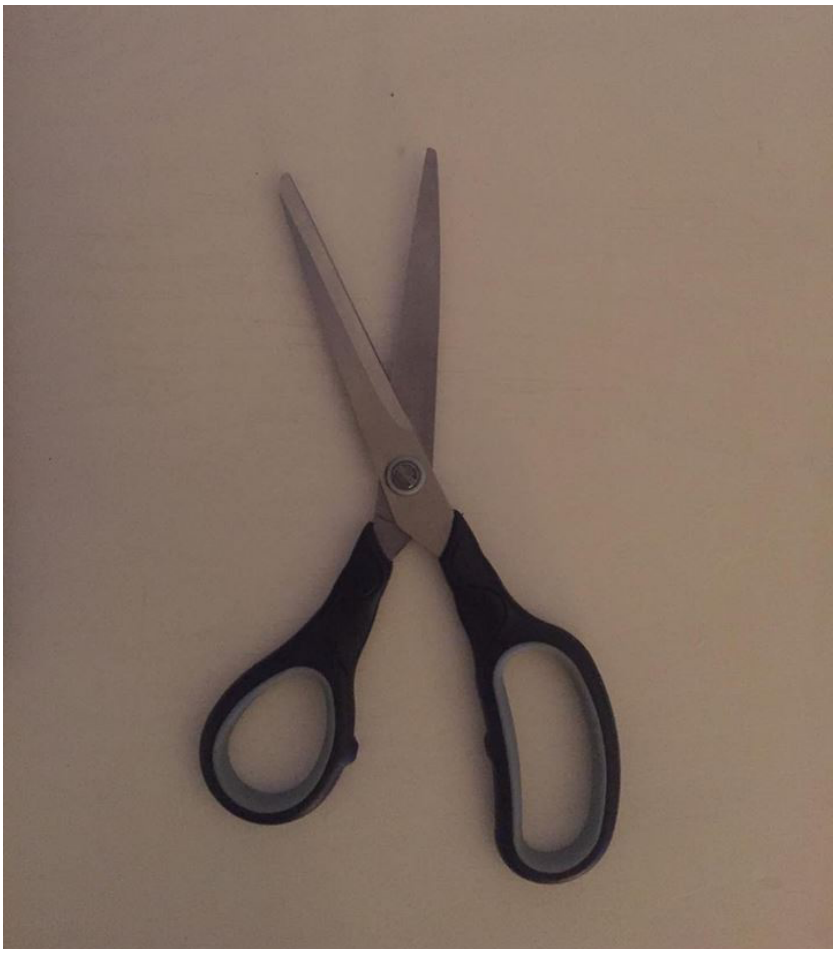


Figure 2.1: A familiar tool showing how mapping can be achieved by having different hole sizes that fits a specific amount of fingers each. Similarly, scissors is constrained to mainly work with fingers inside both holes, and the sharp edges shows a clear affordance of clipping by putting something between it.

2.2 Evolutionary Algorithms and Neuroevolution

Evolutionary algorithms aims at imitating evolutionary concepts from biology, e.g. natural selection, reproduction, mutation, survival of the fittest and recombination [9] and formulate them into algorithms that can be used in machine learning. An EA will typically start out with a random population representing a population of candidate solutions, which is evaluated by a fitness function. If those candidates performs well, e.g. by achieving a good fitness score, they can be added in the next population and might even be used in combination with variation operators (recombination and mutation) to create diverse and novel offspring. This process is then repeated until a specified threshold is achieved.

Neuroevolution (NE) on the other hand, is an EA specialized in evolving artificial neural networks (ANN). Many systems have been developing different methods and ideas on how Topology and Weight Evolving Artificial Neural Networks (TWEANS) should be implemented [30], leading to successfully solving a double pole balancing task [10].

While EAs and NE can be implemented with a focus on specific evolutionary components (e.g the mutation mechanism), this project focuses mainly on the evaluation function (IEC) and the representation of the solutions, and leaves these aspects of the EA to the projects main evolutionary algorithm: Neuroevolution of Augmenting Topologies (NEAT).

2.2.1 Neuroevolution of Augmenting Toplogies

NEAT is an EA that evolves both the structure and connection weights of an artificial neural network with a direct encoding scheme [30]. The NEAT algorithm is inspired by some of the problems that existed in TWEANS [30] including: The Competing Conventions Problem (see figure 2.2), representing random initial topologies, and protecting innovative structure changes even

though adding innovative structure might cause a decrease in fitness initially.

These three problems was in part solved by looking at natures homology principle, which was translated into historical markings (or innovations numbers), making the algorithm able to keep track of which gene is which during the evolution, allowing for more effective variation operators to be applied on the genomes.

By including historical markings to the genotypes, NEAT can then allow for structural mutation on the ANN, without having to deal with the competing conventions problem, or preventing innovative structure changes that may cause a fitness to decrease initially. Likewise, the NEAT algorithm also embraces minimal dimensionality in terms of structure, enabling the creation of random and initial network structures without compromising the search space of the ANNs.

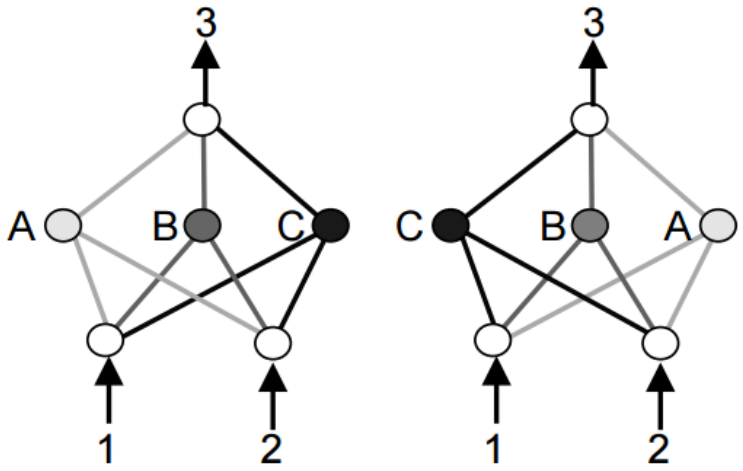


Figure 2.2: The competing conventions problem. Both networks produces the exact same output, even though they have different connections and weights in their structure. This in turn makes them incompatible of producing crossover, and can potentially create damaged offspring. Image taken from [30]).

Since the release of the original C++ NEAT, developers have also produced

a range of new versions of NEAT and its variants. For a full list of the current NEAT implementations see the following footnote[1].

2.2.2 Compositional Pattern Producing Networks

While NEAT is great at evolving the topology of a neural network, the algorithm still struggles on finding structures that can efficiently represent certain patterns through a simple ANN. This can be achieved by using another variation of ANNs defined as Compositional pattern producing networks (CPPN).

CPPNs is biologically inspired from the question on how DNA is able to produce complicated structures like the human body and brain, from very little information [27]. One answer lies in the genotype to phenotype encoding that translates few dimensions into many, by reusing the same information and activating them at any location at any given time. This way of efficiently representing data is defined as *developmental encoding*, where information is unfolding during development, as growth and local interactions is applied [16] to the EA. Another way, is the approach suggested by CPPNs, where layers of information is directly encoded in specific functions that defines the structural relationships resulting from development (see figure 2.3).

[1] http://eplex.cs.ucf.edu/neat_software/

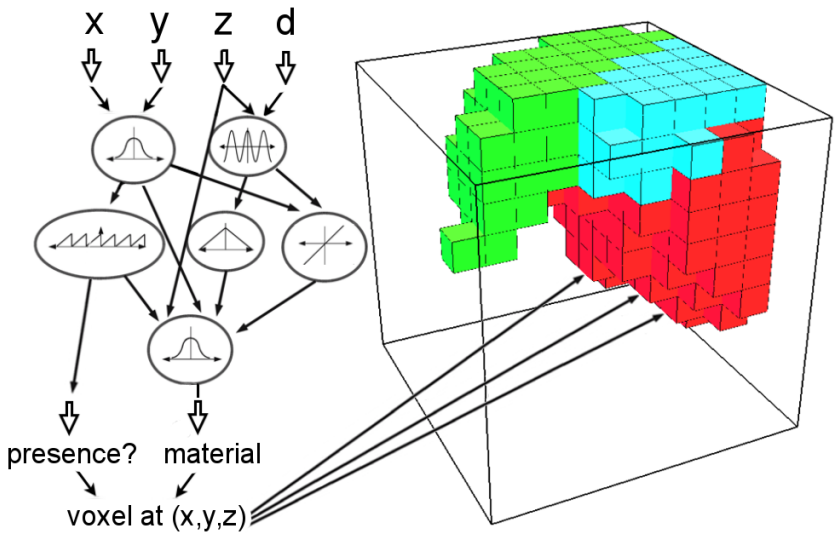


Figure 2.3: Image example of a CPPNs structure and applicability when creating soft robotics (image taken from (http://www.creativemachineslab.com/soft-robot-evolution.html).

CPPNs can be viewed as a variation of ANNs that differ in terms of having an *activation scheme*, and in how they are applied [21] [29] [27]. CPPNs can choose between activation functions from their activation scheme and add them to the hidden node layer of the ANN, making it possible to more easily capture patterns with: symmetry, repetition with variation, regularity and imperfect symmetry [27] [26].

Lastly, CPPNs applicability can be seen as a neural network, that is repeatedly provided with input from a larger data source. For some of the projects mentioned in section 2.5, the CPPN representations would contain the *x, y, z* values or (or just x and y in picbreeder) for each pixel, and the final input representing the distance from the center [24] [24] [7].

2.3 Interactive Evolutionary Computation

Interactive Evolutionary Computation (IEC) is an evolutionary algorithm in which humans takes the role of the fitness function, effectively guiding the evolutionary algorithm by providing feedback to the EA through human evaluations [31]. Users will initially be presented with solutions through an interface, and then select the solutions which is believed to solve a given problem. Offspring for the next generation would then be based on this selection.

This technique allows for subjective evaluation that can be used in creative domains, e.g. where personal preferences on beauty and aesthetics is central to the solution. At the same time, IEC has also been used as an optimization technique [18] [17], and even in combination with other techniques where it can find the stepping stones for potential solutions, or guiding the evolution in deceptive domains[34].

IEC has been used in a wide range of applications of Artificial life, graphic art and CG animation, music, 3D lighting design, editorial design, and industrial design along with many other fields [32].[2]

While these projects certainly underline the potential of IEC, this project has especially been inspired by applications that used IEC in terms of: game playing mechanics [28], creating effective game controllers [14], evolving AI behaviours [25], procedural content evolution [13] [21], generating pictures [24], and generating 3d shapes [7].

2.3.1 The Mapping Relationship in IEC

In order to get a better understanding of how the user believes the inputs is processed and modified by the EA, i.e how the users perceive the relationship between what they select as solutions and what they recieve from the system output, and similarly how the EA interprets the user evaluations and tries to

[2]For a detailed list of conducted IEC projects before 2002 see [31]

produce outputs that matches the input, it is useful to look into the mapping relationship of the IEC.

In sum, the EA tries to produce solutions that matches the *target solution* in the users *psychological space* by repeatedly mapping the user selections in the EAs parameter space. Thereby the output from the EA becomes the fitness axis of a feature parameter space in which the EA searches for global optimum in the IEC system [31] (see figure 2.4).

This means that a well defined mapping relationship can improve the efficiency of which the EA produces solutions closer to the users goal in the psychological space, but also that a poorly mapped relationship can cause misunderstandings, failures, and annoyance potentially increasing the distance between the system output and target solution. This becomes a great issue in terms of human fatigue since users will have to spend more time on evolving the solution.

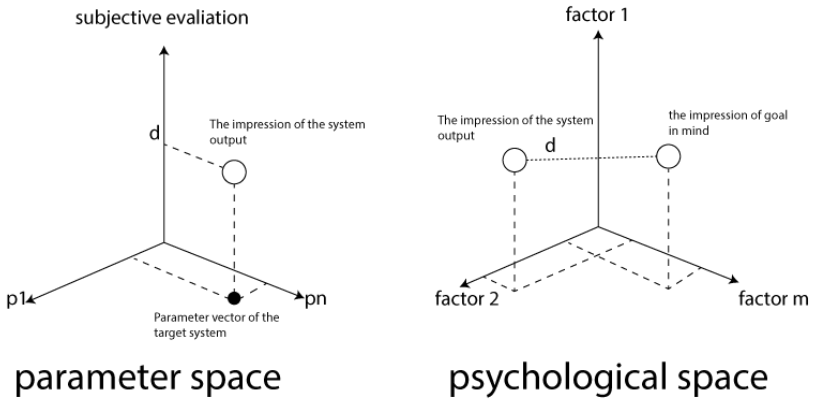


Figure 2.4: Illustration of the mapping relationship based on [31] where the goal for interactive braid evolution is to minimize the distance d as much as possible.

Initially, the mapping relationship may not always be clear to the user, especially if they have no experience with IEC or knowledge of the solutions that is produced. Likewise the first solutions may be random, varied, or

predefined solutions, with no knowledge of the users target solution while the users impression of a target solution may not have been formed yet.

This may create rough and unintended solutions during the first evaluations, but as more and more evaluations is performed, the user should hopefully learn the mapping relationship, while the EA will recieve more and more data to map into the EAs parameter space, i.e. the distance between the system output and users intended goal begins to minimize, ideally because of a more experienced mapping relationship between the system and the user.

It should also be noted, that the target solution may not always be static, and varies from domain to domain, e.g when evolving a picture, users might not have a specific idea of which kind of picture they want to evolve, while assisting an AI controller to navigate out of deceptive mazes has a predefined goal.

2.3.2 Human fatigue

While IEC can provide means of applying human intuition, creativity and subjective evaluations into applications, IEC struggles severely in performing hundreds of evaluations each second similar to a traditional fitness function that utilizes modern processing power in todays computers. In fact, IEC suffers severely from user fatigue, making the amount of evaluations very limited [31] [18] [17].

This has led to several projects exploring different approaches for negating human fatigue. One approach is to make the user perform the evaluation for every nth generation, replacing the evaluations in between with a fitness function, thereby making the user act as an supervisor [18]. Another approach is to create predictor functions that tries to predict what the user might have selected, by using different classifying algorithms [17].

While these approaches provides different ways of dealing with human fatigue, it should be noted that human fatigue is also influenced by the domain in which it is applied to. For instance, human fatigue may be influenced if

the IEC is used as a central game-playing mechanic for a video game, or to figure out an optimization problem in fabricating building facades.

Likewise, the user interface and interaction components used within the IEC application should be taken into account, when addressing human fatigue and when trying to define the mapping relationship. For instance, should the IEC application allow for multiple selection, contain sliders that modifies the amount of novelty in the next generations offspring, or should users also have the possibility to chose which candidates that they did not like at all.

While introducing new interaction components in the interface expands the tools for both the user and EA to create and enhance the mapping relationship of the IEC application, these tools needs to be designed and implemented carefully, expressing a clear functionality and purpose, in order to not be confusing and misleading to the user. This can then be analyzed and evaluated by looking at the signifiers, feedback, constraints and affordances of those tools, in order to better understand the user interaction of those tools. This in turn can potentially make the user aware of what the system can achieve, and how to achieve it.

2.3.3 IEC Projects and inspiration

This section highlights some of the IEC applications that shows different ways of addressing human fatigue, establishing a mapping relationship, and different ways of representing the interface. Similarly, each of these IEC applications has been applied to different domains and showcases how extra functionality can be added to the application, in order to extend the possibilities of IEC.

As noted before, IEC can make evaluations based on a users personal preferences, leading to several applications exploring how pictures, 3d shapes, music and game controllers can be evolved, which reflects the users personality and preferences.

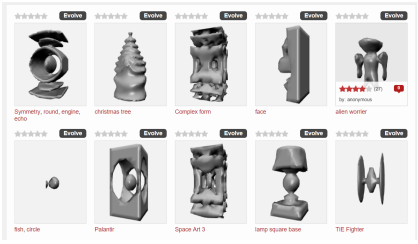
Perhaps most famously, the picture application Picreeder lets users evolve pictures that starts out very simple, but can be evolved into more complex shapes through IEC [24]. Here, users select images they find interesting from a series of images by clicking on the mouse, and those selected images are then used in order to create new offspring for the next generation of images.

While Picbreeder is a great demonstration of what IEC can achieve, Picbreeder also introduces collaborative IEC, meaning that users can evolve solutions collaboratively by branching out from an already published image, and evolve it in a new direction. Likewise they can share and rate evolved pictures on the website. This provides a variety of interesting affordances to the application, which can be useful when trying to engage the user further in the application.

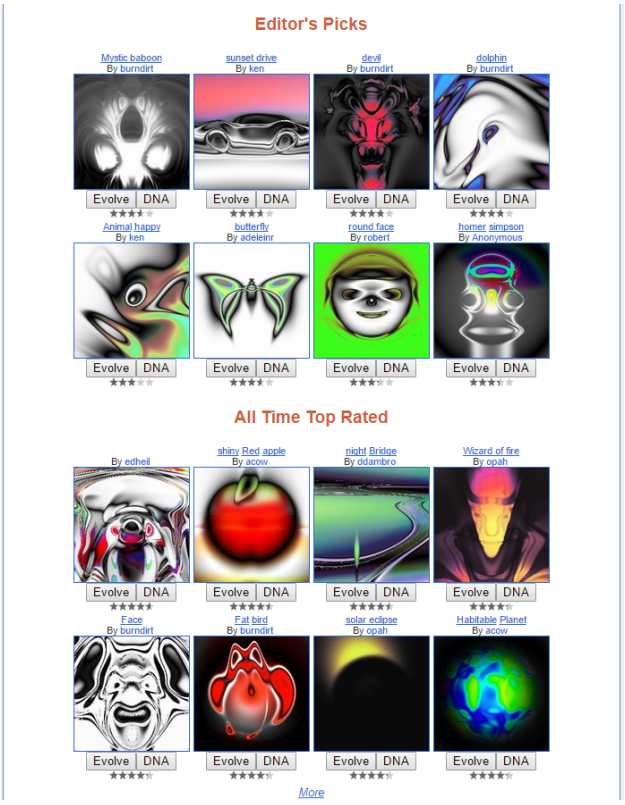
Endless forms is another IEC application exploring the domain of 3d art, with a similar process and interface compared to the picbreeder application. Instead of pictures being evolved, endless forms evolves 3d shapes through IEC (figure 2.5c, providing inspiration on how to achieve interesting 3d shapes that is evolved from scratch by using IEC. Another interesting project building on top of the endless forms engine, lets you upload any 3d model which can then be evolved further [6], instead of evolving it from scratch. Similarly, endless forms also provides a framework on how how certain graphics programming algorithms can be applied to the evolved 3d shapes.

Looking at IEC applications in a social context, the social Facebook game Petalz explores also how user evolved content can be created in terms of evolving flowers [21]. While Petalz also illustrates technical details on how to represent flower patterns by using NE, it also explores how IEC can be used in a social setting with a commercial free-to-play model, motivating users to evolve flowers in order to earn virtual coins that can be spent to buy new species of flowers. These flowers can also be 3d printed and be delivered to the users homes.

(a) The Petalz facebook game



(b) The endless forms interface



(c) The picbreeder interface

Figure 2.5: Some of the inspirational IEC projects using neuroevolution. (a) shows the Petalz interface (image taken from [21]), (b) the endless forms interface (image taken from www.endlessforms.com) and (c) a list of the top rated pictures evolved with picbreeder (image taken from picbreeder.com).

2.3.4 Neuroevolution in video games

video games and artistic projects serves as an excellent testbed for the research field of AI [22], and thereby for the exploration of the capabilities of neuroevolution. This includes experiments that explores optimization problems such as steering cars[33], creating car controllers, [3] [4], or controllers playing Atari 2600 games [14][15].

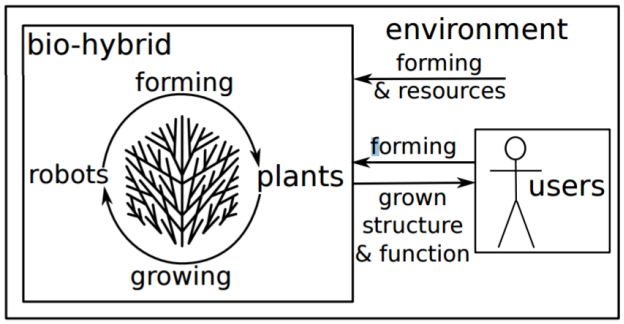
Furthermore, video games has also been created where neuroevolution surrounds the core gameplay, including the videogame NeuroEvolving Robotic Operatives (NERO) [28], where players evolve armies that compete against each other, or the Galactic Arms Race (GAR) where users controls a spaceship and evolves particle weapons by using the weapons the users found interesting [13] [3]. These games serve as inspiration on how IEC can be used in combination with neuroevolution as central game playing mechanics.

2.4 Flora Robotica and the social garden

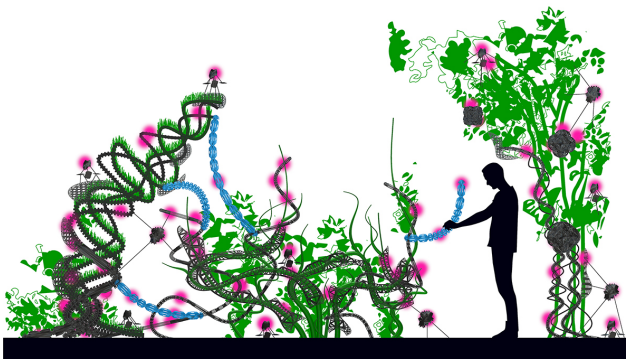
Flora Robotica is a EU funded research project exploring robot-plant biohybrid systems [12], across 6 different universities in the EU. It investigates how plants and robots may live together creating symbiotic relationships with each other, extending the functionality of biology and robots (see figure **??**)

Figure 2.7a illustrates some of the main concepts behind Flora Robotica, from the beginning to the end in a closed loop with different processes. Here, forming and growing are two major processes that can be handled differently and with different approaches (e.g. the growth model of a plant can be created by different machine learning techniques with data from a specific plant species). While this project only focuses on the *forming* process, in

[3]For a more detailed explanation of the current state of neuroevolution in video games see [22] for a overview and more detailed explanation



(a)



(b)

Figure 2.6: Two images describing the Flora Robotica project briefly. (a) shows the a general model of the main concept behind Flora Robotica, while (b) shows an example image of the social garden.

particular when humans are included in the forming process through an IEC application, it should be noted that this project also strives towards working together with other processes that can achieve the growing and fabrication step, and as such, as able communicate with other processes developed in the Flora Robotica project.

Another aspect from Flora Robotica is the social garden [12], where humans can interact with the bio-hybrid systems, e.g their behavior patterns and structures, and then be able to form, modify, and maintain a social garden as illustrated in figure 2.6b. Two in-world examples on the social garden concept can be found in figure **??**

2.5 Braiding and complex modeling in architecture

Since it is out of the scope for this project to describe how fabrication and braiding techniques can be applied in order to create braided structures in depth, only a brief overview of the literature will be provided. Braiding and representation of weaving patterns can exists in many different variations, and can be represented digitally in many ways as well [5]. On top of that, the material attributes of braiding (e.g wood vs linen) has different implications on the fabrication process, their sustainability, and how their usability.

This also highlights one of the many reasons why braided structures was chosen as a point of origin for evolving bio-hybrid structures from an architectural perspective at CITA. First of all, braiding can create many different structures that can be morphed, deformed, and scaled while still containing enough high-level architectural information to be analyzed. Likewise, they are sustainble in many environments, and allows for the integration of sensors and actuators during and after the fabrication process.

(a)

(b)

Figure 2.7: Two images showing examples of fabricated braided structures that is inhabited by plants. Pictures used with permission from CITA.

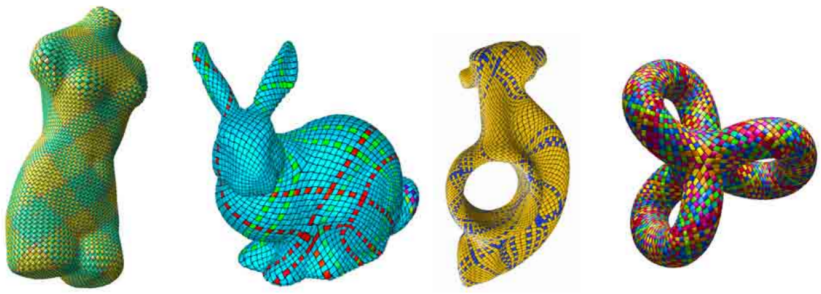


Figure 2.8: Different weaving patterns on a range of mesh objects. Each object is constructed with ribbons and has different cycles of cyclic plain-weaving applied to them. Image is taken from [2].

3

Application overview and data structure

The overall goal for the project is to explore how users can evolve different braid structures through an IEC application that uses CPPNs, and how different and novel interfaces and input devices can be used in order to increase the efficiency of the application.

In this section, focus is shifted towards the technical aspects of the IEC applications functionality. Here, a technical overview of the applications processes and their communication protocol will be described, along with the different CPPN representations, and a step-by-step description on how braid structures are created and modified from an initial list of points to a fully polygonal mesh that is ready to be imported into the simulation process, where users can evaluate the structure.

3.1 The application

This application can be divided into three processes: a process for the simulation step which takes place in the game engine Unity; a process for creating 3d models taking place in the 3d application Rhino with the grasshopper plugin; and the evolution process where the NEAT algorithm is applied. Each of these processes has a responsibility for a certain part of the applications functionality, and is believed to negate the weaknesses found in other processes, or replace the functionality for certain operations. To gain a better

overview of the processes and their connections see figure 3.1.

For instance, while Rhino can model, analyze and edit 3d models, it is rarely used to create a user friendly environment for simulating animation, sound, and other visual effects that reacts on user input. Unity (or any other game engine) handles this excellent since those techniques are very common in video games, but Unity is rarely used for modeling complex structures and analyzing 3d meshes for their physical properties in which Grasshopper can be used.

3.1.1 The connection protocol and application loop

The processes uses a local UDP connection to send JSON packages back and forth. These data packages contains information of the application state and modeling information of the braided structures. The package contents are dependent on which process they are sent to, e.g Unity sends much more modeling data to Grasshopper, while Grasshopper simply sends a reference to the 3d modeling file to Unity. The packages sent between the processes is also dependent on the current application state.

For instance, Unity will not query a request to model additional structures if Grasshopper is currently modeling. This state checking can hinder internal process errors, which can potentially crash the application, and might lose important information on the structures that the user was currently evolving.

Therefore, keeping track of the application state should always be done before sending larger packages between the processes. Also, sending packages between processes requires little effort for the application, where unwrapping large package contents can demand much more more effort.

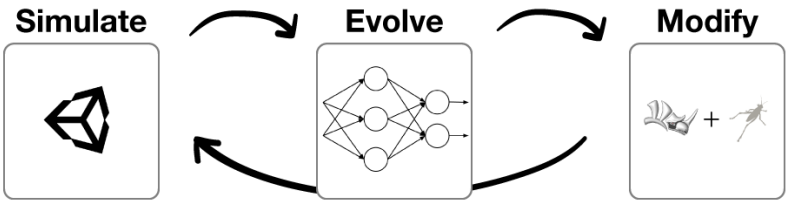


Figure 3.1: Overview of the applications processes and their connections. The simulation process uses the game engine Unity (left image), the evolution process uses the NEAT algorithm and CPPNs (middle image), and Rhinoceros3d and Grasshopper plugin is used by the modeling process (right image). Together, these three processes creates the overall functionality of the application.

3.1.2 The data structure of the braids

The braiding structures this project takes inspiration from is based on the braiding prototypes built at CITA (see figure 3.2). These prototypes has a general rounded tall structure with a relatively small diameter, but have been created in more complex shapes in various sizes, with different braiding patterns and different material densities as shown on figure 3.2. Their overall shape can be boiled down to cylinder shapes, which is biased in the initial data representation.

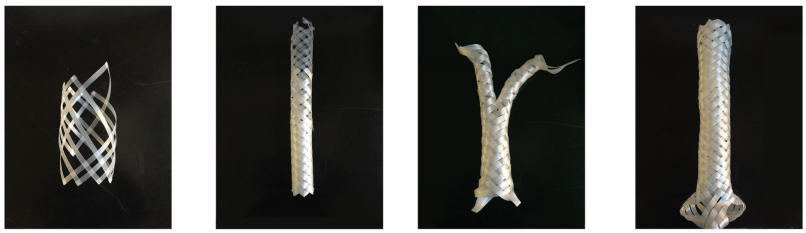


Figure 3.2: A small sample of some of the handmade braid prototypes created at CITA studio. Each prototype tries to explore some of the possibilities of either the structure, material, or braiding pattern that exists in braided structures.

The braids is represented through data trees, where each node holds an id, radius value, a vector, and a depth value. Each data tree then holds the recipe of that particular braid, while a node holds information on each segment of the braid. This data structure further allows the addition of values to each segment, making it possible to expand the data tree with other parameters.

3.2 Representing the braids to the CPPN

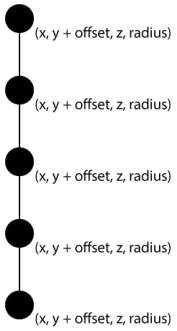
Initially, a braid consists of a data tree consisting of n nodes, each node being given the parents y value plus an offset value. This creates an initial braid with a cylinder form (as seen in figure 3.3a).

Next, depending on the setup, each node on the data tree provides input values to the CPPN, which in turn provides values that is used to modify that particular node (figure 3.3b). Then, the data structure is formatted to a JSON object, containing an id and lists of vector values, which is sent to the modeling process, which performs a modeling algorithm that creates the braided structure based on the JSON objects lists of vectors (see figure 3.3c). Finally, the braid structure created by Grasshopper is exported and imported in Unity at runtime, where it is mapped to the NEAT algorithm and is ready to be evaluated by the user. For a video showcasing the applications processes see application-showcase.wmw.

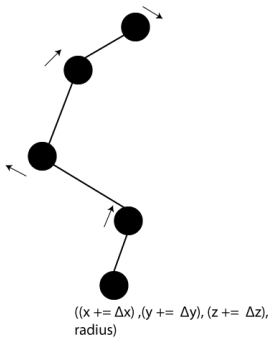
3.2.1 The CPPN representations

The first initial CPPN setup received one input and had two outputs. For each node on the data tree, the network was provided with the nodes y value as input, in which the CPPN outputs would be added to the nodes x and z value. This process is then repeated for each node in the data tree.

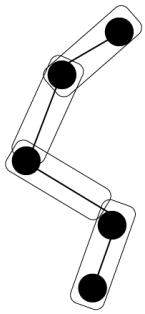
The first braids created with the simple CPPN setup served as the initial proof concept for testing out the application, and ensured all of the appli-



(a) Initial list of nodes.



(b) Modified list of nodes.



(c) Modeled list of nodes.

Figure 3.3: A storyboard of the basic steps for creating braids in interactive braid evolution. Initially, an empty list is created with an offset as seen on image (a). Each node is then input to the CPPN which is added to each node (b). Finally, each node is been used by the modeling step to create the final 3d mesh of the braid structure

cations processes were working as expected. Even though these braids were based on the simplest representation thought possible, the braids were still varying in structure to some extent but with identical radius values and without the possibility to create branches.

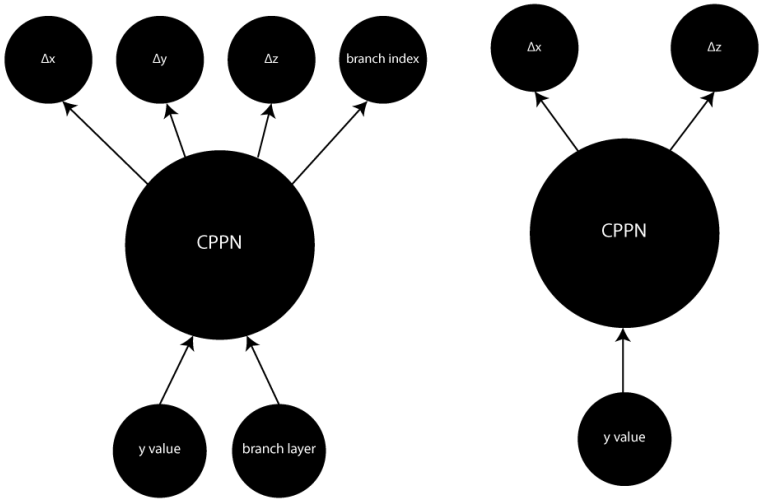


Figure 3.4: The projects two main representation of CPPNs. The left side illustrates the final CPPN representation, and the right side illustrates the initial CPPN setup.

The next CPPN setup focused on extending the structural possibilities of the previous CPPN setup by adding a new recursive layer to the evolution process that would allow for creating braided structures that was able to split. braided structure that could split. This extended CPPN setup would receive two inputs, the y value of the node like the previous setup, and the *branching layer* of the node. The output layer of the CPPN would be extended to now contain four outputs: delta values of x, y, z that would be added to the nodes vector value, while the last value defined as the *branching value* would determine whether a node should branch if above a certain threshold (see figure **??** for a storyboard when creating braided structure that would allow for split).

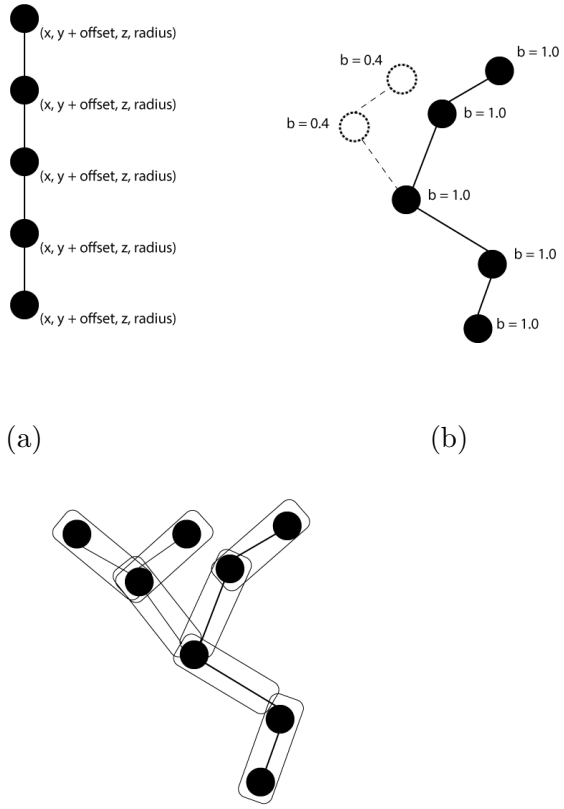


Figure 3.5: A storyboard of the basic steps for creating braids with splits. Like previously, an empty list is created with an offset and is input to the CPPN (a). If the branching value is higher than a threshold a new list of children is added to the node (b). These new nodes can branch as well, and is likewise sent to modeling step that creates the final 3d mesh of the braid structure.

If the node should branch, a new series of child nodes with a different branching layer would be added to the node and then input to the CPPN (figure 3.5b). These child nodes would likewise be inputted to the CPPN and would also be able to create new branches. This can lead to potentially many branches, especially if leaving the option open for multiple branches, which lead to the introduction of a branching threshold, which would be reached if too many branches were created in the data tree. Furthermore, if a node was branched, the radius values for the nodes children would correspond to half the value of their parent nodes radius.

For this project, the amount of children would be equal to the length of the initial branch size (i.e the amount of nodes attached to a branch) minus the current depth of the node that the new list of children was going to be appended to. This makes the amount of children added to never be greater than the amount of children in the parent node. However, the CPPN can still provide delta values to the new branched nodes y values, making the CPPN make new branches that in theory could appear longer than its parent branch.

4

Simulation and interface

In this section, focus will be directed towards the design aspects of the IEC application. First a framework for analyzing the simulation process is provided in order to get a deeper understanding on how the interface and input devices can be used in order to address human fatigue and establish an effective and flexible mapping relationship.

This framework will be used in order to explore how AR can be integrated, but AR in IEC will likewise be explored by looking and comparing the applications two interfaces which is accompanied by two different input devices. The first interface is focused on using a keyboard and mouse as input devices, while the other interface is focused on using the LEAP motion controller as input device. These interfaces has different possibilities and design challenges in each of the simulation steps because of their input devices, but nevertheless requires a good conceptual model so the user can focus more quickly on learning the mapping relationship and use the IEC application efficiently.

4.1 The three simulation steps

As mentioned above, the simulation process can be divided into three independent simulation steps that occurs chronologically each time a trial is performed by the user. Each simulation step represents a specific application state in which the interface and possible user interactions is changed. While some of the simulation steps has transitions which from a technical view can be hard to distinguish from (e.g. when models is created and imported si-

multaneously), it is possible from a interactionist viewpoint to more easily distinguish between these simulation steps (see 4.1 for a full overview).

The *evaluation step* is where the user is presented with the CPPN outputs and tries to evaluate the properties of these outputs, in this case the braid structures. In this project, this is done by looking and moving around braided structures, and in rare cases by presenting vector information on each braid segment. This step holds potential for further exploring how additional affordances can be integrated to provide the user with more freedom for performing the evaluation step, e.g. by integrating inspection of the raw values of each braid segment.

The *selection step* is where the core user interaction takes place. This is traditionally done by clicking on an object and requires special attention to the feedback and signifiers communicating the actions that user is about to perform, and which objects they are going to be applied to. This is likewise not restricted to visual effects, e.g. hovering or adding a border around to the structure, but can just as well be communicated through audio channels, shortcuts or something third.

Finally, the *modeling step* is where the user waits for models to be created and imported into the simulation engine. This is where most of the waiting time occurs and most of the user interactions is disabled. To negate waiting time, several actions can be taken. Initially, the user can be provided with information on how much waiting time is left (similar to a loading bar), or perform other interactions that is unrelated to the IEC process in general (e.g. flying around or playing with the hands). But in sum, this step, should be minimized as much as possible since it is a main contributor for creating waiting time, and thus in creating human fatigue.

Figure 4.1: The three simulation steps: evaluation, selection and modeling step, in which users interface and possible user interactions with the IEC system is changed in each step. These steps are encountered in a specific order for each trial.

4.2 Setting up a scalable interface

The users of interactive braid evolution is expected to have little or no experience with EAs. Therefore the initial setup of the interface is very sparse, constraining all actions except for clicking on a button and moving the camera with the mouse (see figure 4.2). This also ensured the initial amount of learning required to perform each simulation step was kept at a minimum.

Figure 4.2: The first initial interface of interactive braid evolution. The right windows contains messages recieved from other processes, network log, and internal messages. The sliders on the left side controlled variables for the modeling algorithm, and advancing the simulation step could only happen by clicking on the button in lower middle of the screen.

By using a scalable approach, it is also implied that the interface would be repeatedly evaluated and improved during development, based on several user sessions (described in the experiment section). These sessions would evaluate the use of each interaction component in the interface, specifically any errors that would occur from mapping, affordances, feedback and signifiers. Also, as the users would familiarize themselves with the application and learn how to use the interface, new ideas and improvements to the used interface would be discussed. Additionally, some of the applications processes got optimized and extended functionality during development which would also impact the interface design.

4.3 Learning how to use the application

Since users does not have any experience with either IEC or EAs, failures and confusions between the user and the system will inevitably occur, but this can be remedied in several ways. First, tutorials or manuals on how to

use the system can be provided describing the application before the user begins to interact with it. Secondly, creating the right feedback, constraints and signifiers about each operation can likewise avoid failures, and lastly: by integrating operations that can modify or reverse previous operations.

However, it can be too much information to learn all of the possible operations simultaneously, making it important to plan when and how those operations should be introduced. Furthermore, users should always be able to seek information about the application and its operations, especially when having multiple operations that needs to be memorized. Communicating how to access this information, while also ensuring that the information provided can answer the questions asked by the user, is likewise of vital importance when learning how to use the application.

4.4 Designing the LEAP interface

One challenge of designing the core user interactions with the LEAP motion controller is the lack of experience of using it, compared to the mouse and keyboard which have become a part of the everyday life. This creates new challenges and possibilities that needs to be carefully designed in each simulation step of the IEC application with the user in mind.

Another aspect to the LEAP interface is related to the technical properties of the LEAP controller and the functionality of the hardware itself. Since the LEAP controller is scanning for the hands from the table and upwards, some gestures and fingers can be hard to track since they can be obscured from the scanner (see leap1.wmw and leap2.wmw). Likewise detecting the angle and distance of each finger joint can be imprecise if the fingers overlap each other, making it hard to detect which fingers is stretched and which is not. On the other hand, the direction and position of each hand palm is precise, but also influences the interpretation and precision of detecting the fingers. Lastly, in rare cases other objects can be interpreted as a arm or

hands which can also create unexpected behaviors.

These technical implications had several impacts on the final design of the leap interface. First of all, hands should be encouraged to be away from each other with a proper angle to enhance the precision when scanning for hand gestures or if fingers is stretched. Secondly all fingers should be evaluated when looking for hand gestures, since one finger might be obscured for the leap controller and is hard to track. This is important since the operations that is performed can be irreversible (e.g advancing the generation) but has to be simple enough to be memorable for the user.

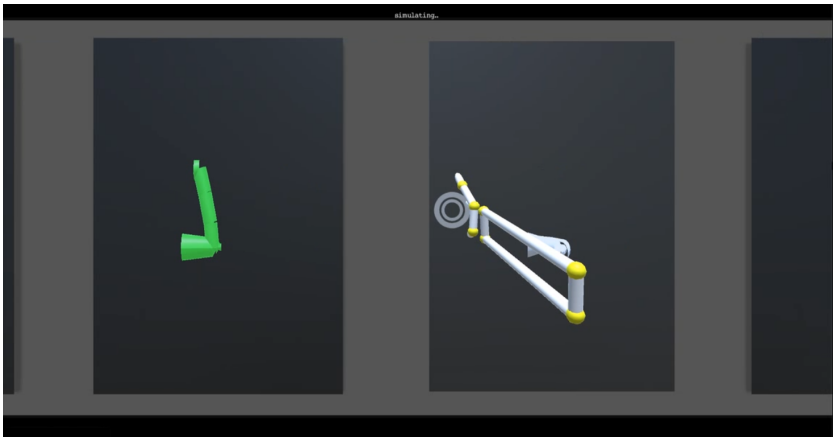


Figure 4.3: The LEAP interface, where braids was selected by touching ui elements with the hands, and browsing through a collection of braids by using swiping.

To overcome some of these obstacles, several constraints and changes was introduced to the interface (based on the user sessions described in appendix A). First of all, the population would be showcased as a collection of braids which could be browsed by the user (see figure 4.3 and the video leap-interface.wmv for a walkthrough of the interface).

This collection could be browsed by swiping, touching and poking which would become the main ways of interacting with the simulation. To further constraint the user, only the right hand would be able to swipe and select

the braids, while the left hand could stop the dragging by creating a fist gesture For advancing the generation, the hands would need to be close to each other. The main principle for isolating the right hand to interact with the ui components was to constrain and separate the functionality into each hand, i.e the right hand would only effect the simulation by interacting with the ui components through touching, while the left hand would only affect the simulation through hand gestures.

5 Experiments

In this section, the evolutionary parameters and CPPN representation will be described for the projects two main experiments that was conducted at CITA studio. These experiments provides results that will be used to discuss how IEC can be used in the field of architecture in terms of creating braided structures, but also in order to explore the potential of using AR to increase the efficiency in the field of IEC. The first experiment is based on the keyboard interface while the second experiment is based on the leap motion interface.

While these experiments varies in terms of their interface and input devices, they still contain identical CPPN setup and NEAT algorithm parameters, making them able to evolve the same type of braided structures. Finally, the methods and approach behind the user sessions that was used to develop the final two interfaces is described in this section as well.

5.1 User sessions

The development of the application was based on a user-centered [1] and participatory design process [23] by first creating a minimal and scalable interface for advancing through the simulation steps. This approach focuses on involving the user in the design process, by observing and discussing how users were using the application, and how they would like to use the application. Users would then become co-designers of the application, providing feedback and actively contributing with ideas on how to further improve the

design of the application. Unfortunately, these sessions were rarely recorded, but most of the relevant details for each user session was written down as notes and can be found in appendix A.

The focus of the user sessions was not limited to only test the usability and interaction with the application, but could explore other aspects of the application as well. This included user sessions that was specifically investigating how users felt their input was recognized by the CPPN, how the evolutionary parameters could be tuned, how the braid was represented and how new functionality could be added. This exploration could likewise change during the user sessions, and several aspects was often explored during the same user sessions.

Getting a better understanding of how users learned to use the application was approached similarly by taking notes and having discussions while users were currently learning how to use the application. Specifically, users would be asked what they felt hardest to understand or confusing to use. This would then lead to a discussion where feedback was provided that could be used to further optimize how the learning process should be designed for the application.

5.1.1 User sessions for the LEAP controller

user sessions directed for testing the leap interface were likewise based on observations on how users interacted with the LEAP motion controller, and by having regular discussions on how a intuitive and natural interface could be developed. However, users would also be tasked to move, select, and rotate a 3d cube model by using the LEAP motion controller, but had to figure out how to do it themselves, when in fact no gestures or hand movement had been implemented yet, even though users could see their hands interpreted into the simulation. This in turn created a context for observing how users were intuitively using the LEAP controller without any guidelines or instructions, and which actions users would perform in order to figure it out how to solve

the assigned task.

5.2 Evolutionary parameters and CPPN setup

The initial population would contain half connected CPPNs with at least one hidden node with a random activation function. The activation library would contain of a *Linear*, *BipolarSigmoid*, *Gaussian*, and *Sine* activation function. Elitism would be enabled to ensure at least one of the users selected solutions would survive to the next population.

For the NEAT algorithm, the asexual proportion would be set to 0.6 and the sexual proportion to 0.4.For the genomes, the mutation probability for adding a node was set to 0.6, deleting a node to 0.3 while adding a connection was set to 0.15 and deleting a connection was set to 0.1. Mutating weights was set to 0.2, and the weight range between -2 and 2. This setup was created in hope to create as much variation in the population as possible, leaving room for novel structures, while still holding a resemblance to the previous user selection. Also, leaving room for novel and very different structures makes it possible for the user to seek other directions to guide the evolution towards.

5.3 Braid experiments

Each experiment would be initiated by an optional choice of watching a brief tutorial describing the overall goal, context and simulations steps of the application: evolving braid structures through IEC that can be integrated in bio-hybrid systems similar to the social garden. The concepts and descriptions was presented through images, other evolved results, and animation, and each simulation step was explained through animation clips, showcasing exactly how each step is performed.

However, the tutorial does not try to explain the mapping relationship of the EA, in which it was estimated that users should instead try to learn it

through experience and trial-and-error. Therefore, Users would first evolve with the author or alone for a couple of generations, before doing the experiment, ensuring that the user knew how the application worked, and how to proceed between the simulation steps.

The CPPN setup for both experiments is the same as previously described in section 3.2 with 2 inputs (branch index and y value) and 4 outputs (delta x, delta y, delta z, and branch value), with radius values starting at 2.5, which would be halved for each braid split. The splitting threshold would be set to 0.8 meaning that any CPPN output for the branching value above 0.8 would create a split in the structure and add new child nodes to the braid.

Users would also be randomly picked from the CITA studio and was told they were free to evolve whatever braid structure they found interesting. Once the user had tried the application and watched a tutorial, they would first perform the keyboard experiment for at least five generations or more, and then optionally asked if they wanted to try the LEAP experiment for at least five generations as well. During the LEAP experiment, some users would also be recorded. As in the keyboard experiment, no objectives to what should be evolved was given to the leap part of the experiment.

During the keyboard and LEAP experiment, each braided structure would be saved, along with screenshots of the braids, author name, braid nicknames, and the amount of generations performed.

6
Results

A total of 304 braided structures were evolved in total during the experiments from 9 different users, in which 234 braid structures was evolved from the keyboard experiment and 70 braid structures was evolved from the LEAP experiment. A total of 9 keyboard experiments were conducted while 5 LEAP experiment was conducted. These results have been used to create several storyboards showcasing how simple braid segments can effectively be evolved into interesting and subjective structures. Furthermore, the storyboards can be used to describe how various user strategies can be applied in order to achieve desired solutions from the EA.

The keyboard experiment would have an average of 8.8 generations and 26 evolved braid structures per user, while the LEAP experiment would have an average of 6.4 generations and 14 braid structures per user.

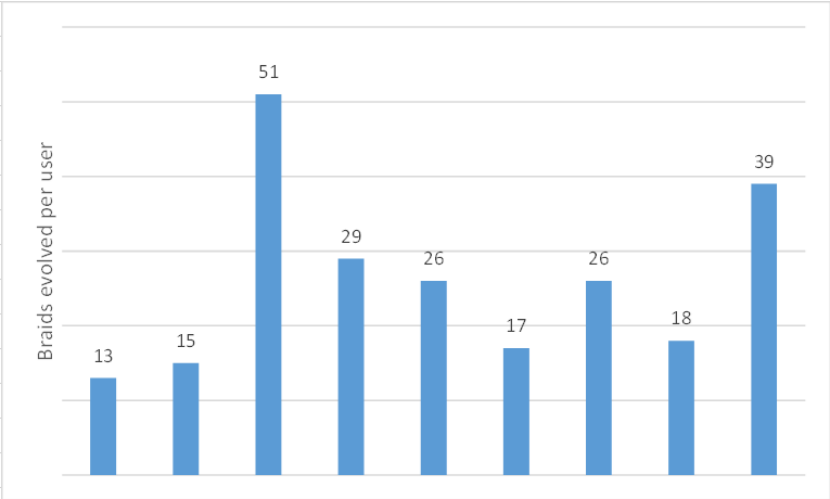


Figure 6.1: A diagram showing the amount of braid structures created for the 9 keyboard experiments

Also, 12 user sessions have been noted and added as appendix A, which have all contributed to improve the applications two interfaces. Two videos are also attached to this project showcasing some of the challenges when using novel input devices for IEC.

6.1 The braided results

Three storyboards is included: One based on the keyboard experiment, the second on the leap experiment, and a full storyboard showing an experiment with more than 5 generations. While the angle of which the pictures was taken can potentially obscure some of the structure, the models was tried to be positioned in a way where the structure could be showcased best.

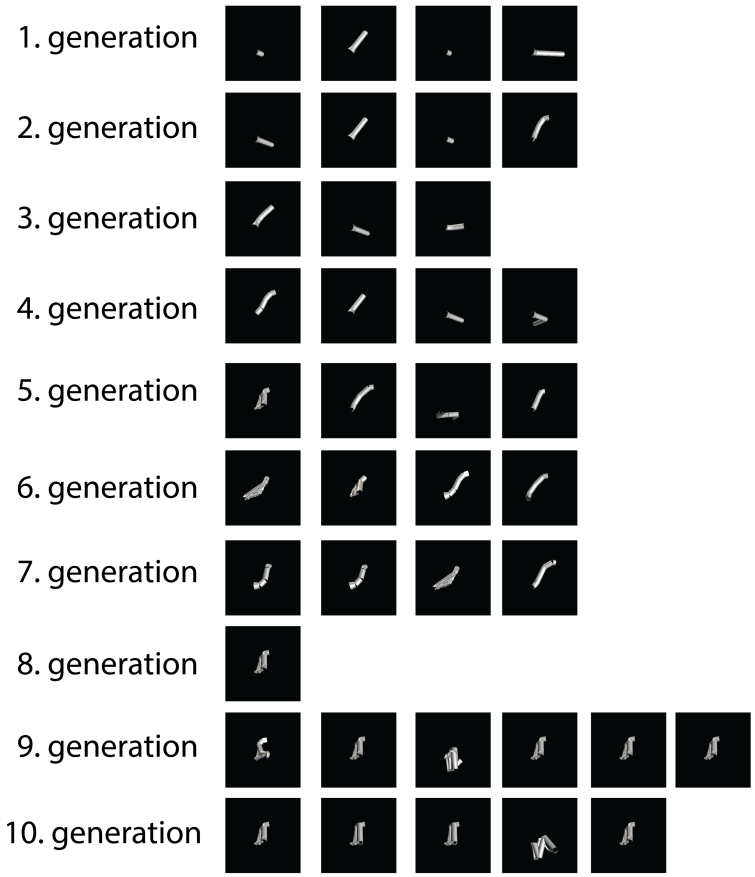


Figure 6.2: Storyboard of user evolved braid structures for 10 generations. Each generation shows the braided structures that was selected by the user.

Figure 6.2 shows that complex braided structures begins to emerge from simple structures after the 4th generation. From the 4th generation to the 7th generation, braids with s-shapes is chosen, and from the 6th to the final 12th generation, braid structures with some kind of branching integrated to their structures begins to dominate the selection. Another interesting thing to note, is the amount of selected braids for each generation. Almost half of the population is selected for each selection step, except for the 8th

generation where only one braid was selected, while at the 9th generation, over half the population was selected.

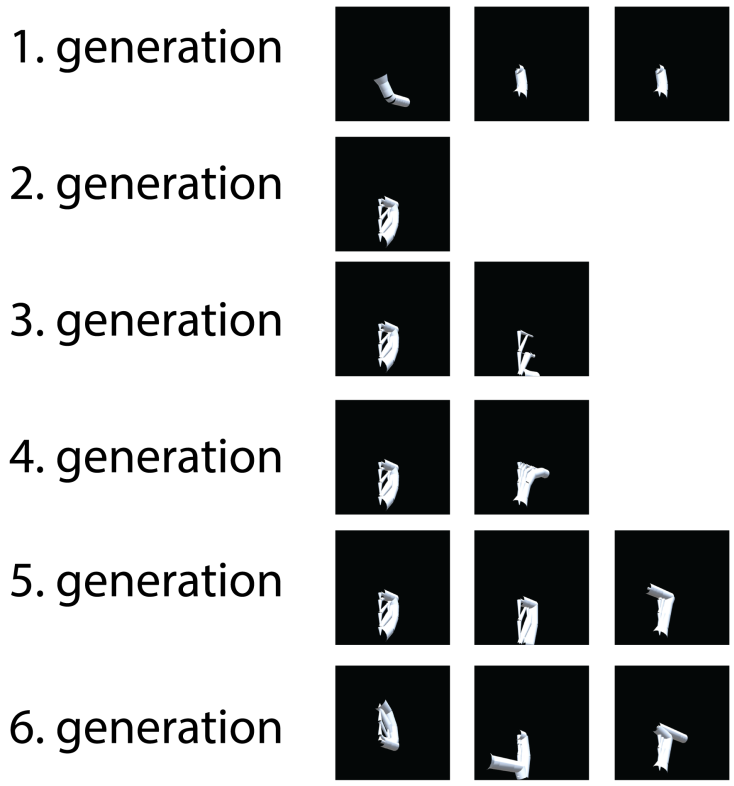


Figure 6.3: LEAP storyboard of user evolved braid structures for 6 generations

Figure 6.3 shows several interesting patterns in terms of the results achieved at the 2nd generation, and especially in terms of the user selection in which the same braided structure is repeatedly selected for three selection steps in a row.

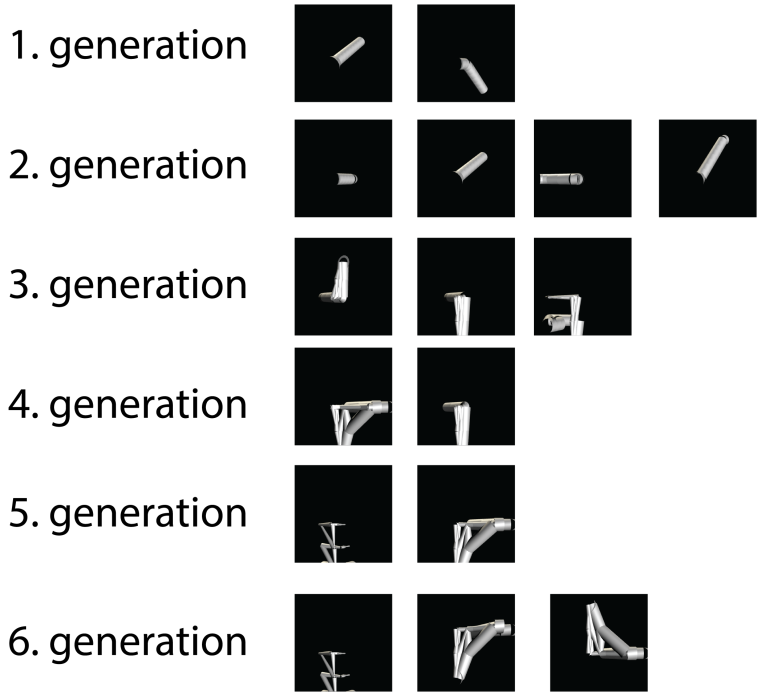


Figure 6.4: Keyboard storyboard of user evolved braid structures for 6 generations

Figure 6.4 shows how complex and branched structures is once again preferred by the user in later generations. This time, branched structures is evolved after the second generation, but another interesting pattern can be observed from the 3rd to the 5th generation. Here, the user selects the braid structure which was previously selected from the last generation, and combines it with a new braid structure. This happens from the 3rd to the 4th generation, the 4th to the 5th generation, and 5th to the 6th generation. This is particularly interesting, since it can be used to get a better understanding of how the mapping relationship is interpreted by the user.

6.2 Optimizing the interfaces with user sessions

The first interface contained four debugging windows listing information on the user selection, the application state, and other messages received from other processes (as previously seen on figure Figure 4.2). All models would be imported when all models were created, and selection was done through mouse clicking. The camera was restricted to look between 0 and 135 degrees on both axis at any time and a slider was integrated to change the height of the evolved braids in the next generation.

Two user sessions (A.1.1 and A.1.2) along with optimization on some of the applications processes led to the second version of the interface where all debugging windows was removed and replaced by a loading bar. The loading bar would be updated frequently, giving the user feedback on the current application state. Additionally, models were loaded dynamically into the simulation as soon as they were modeled, instead of importing them all at once. This also implies a change in the communication and data structure in which nine individual packages would be send instead of one large data package. Thirdly, visual signifiers and feedback were added on hovering and clicking on a model.

Additional affordances was likewise added to this interface. First a flying ability was added, making the user free to move around with the arrow keys, allowing for better evaluation of the braids. Next, deselection of current selected braids was implemented, along with shortcuts for advancing the generation and focusing camera to population. A menu was likewise added to the interface, listing all the available operations and their respective shortcuts, while also replacing all buttons by shortcuts.

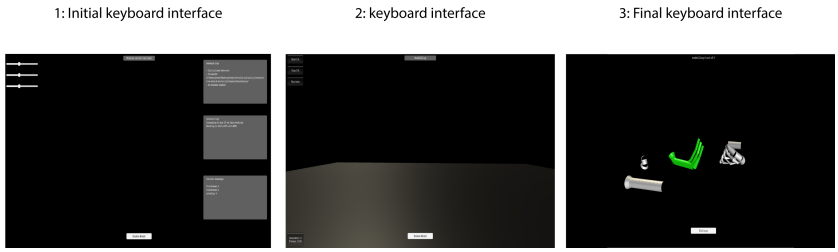


Figure 6.5: The main three interfaces of the keyboard setup from the beggining of development to the final interface used in the experiments from right to left

6.3 The LEAP experiment

Although several braid structures was created with the LEAP motion, it was evident that the LEAP implementation lacked several precision when performing the operations of swiping and selecting a braid (see leap0.wmw, leap1.wmw, leap2.wmw and appendix A). However, a key observation on how the LEAP interface was used, lies in the fact that the hands is visualized in the simulation and provides instant feedback to the user on their hands position. This can also be observed from the fact that in leap1.wmw and leap2.wmw both users have no issues arising for moving the hand towards the braid they intend to mark as selected. Instead, the issue lies when performing the operation that marks a braid as selected.

Another important difference between the keyboard and LEAP interface can be found when comparing the evaluation steps from both interfaces. Usually with the keyboard implementation, users would move to a spot where they could see all of the braided structures they were going to evaluate, compared to the LEAP experiment where users would have to swipe several times in order to evaluate the whole population. Since only three braids can appear on the screen simultaneously, enhanced by the fact that the user had to make a large effort in order to swipe, this in sum made the evaluation step more inefficient.

7 Discussion

7.1 Using user-centered and participatory design in IEC

When developing an IEC application, the interface and processes can be changed several times in response to the achieved results and the stepping stones for achieving them. By choosing a user-centered design process for developing the application and by making users co-designers of the IEC interface, feedback can be applied to each simulation step in a consistent and efficient manner.

For instance, as noted in a user session, a user described that the context of selection should happen in a museum with glass boxes showing each braid, since the space was too wide, and users could sometimes be lost in the simulation by being allowed to fly around. This addition of context could then further constrain the simulation to only take place inside a room. While constraining the user to a room may not necessarily improve the keyboard interface, it hints that the context in which the simulation takes place in can influence the motivation for performing IEC. These details and feedback can be invaluable during development and ensure that the design choices is properly tested and expanded in a healthy way.

Another benefit for having a user-centered approach when applying IEC is in order to verify the various design choices that has been mode during the development of the application. For this project, this is exemplified in the

approach of investigating how AR could be implmeneted in IEC by letting users perform the hand gestures they thought as intutive and natural for certain operations. This furthermore illustrated that while gestures might be good for signaling operations in the application, hand gestures for browsing through a collection of braids might not be as intuitive and natural as first anticipated, which lead to the development of touching on ui elements instead of using hand gestures in the LEAP interface.

Lastly, the project was further enhanced by using this approach in the context of CITA studio. Here, many people would contain key knowledge on how to model, fabricate and design braided structures from different aspects (e.g material, robotics or digitally), which would in turn help optimize many of the applications processes. These users would then function as expert users [8] in which specific and complicated questions could be tailored to, providing more rich and useful feedback depending on the aspects the user session sought to explore.

7.2 Comparing the interfaces

As seen in the result section, the interfaces was changed on a frequently basis and carefully tested before being implemented. While these interfaces vary in terms of how they react to each simulation step and in their input devices, the design solutions to each of them is not always guaranteed to be the right one, even though they were tried and discussed through several user sessions.

For instance, by removing the debugging windows from the final interface, users would have to only rely on the loading bar for displaying information about the application state. Another approach could be to make the messages received in the debugging windows more easy to understand. This could in turn make users aware of the application state and enforce their understanding of the overall application and its processes, aiding the users in understanding why and how their operations would fail or at least not

perform as expected.

While this approach provides a better overview and deeper understanding on the IEC application itself, this would also require a greater effort from the user to learn the application, since they now had to respond to the messages received from the debugging windows, on top of evaluating the actual output braids like they normally would have to do. The design question is then to decide how much information is exhausting for the user, how much of the information is actually used by the user, and how much of the information is relevant in the current simulation state. Based on several user sessions and discussions, it was then decided that the debugging versions did not have a clear purpose when evolving braid structures, they were not used by the user in either of the simulation states, and while they could potentially make the user better understand the application overall, it was estimated that most of the focus should be directed towards learning the mapping relationship instead.

7.2.1 Exploring additional details of the braid structure

When evolving with IEC, exploration of the output solutions should be encouraged in as many ways as possible, without cluttering the interface or demanding too much effort from the user to perform the evaluation. One advantage of adding additional information to each braid structure in the IEC application, is that the EA can provide the user with more information of the output solution, while the user is able to gather more information on why the EA evolves those structures. For this project this is all done visually, but throughout the user sessions several users requested a feature that allowed to inspect the physical properties on the braid structures they had evolved. This makes sense, since fabricating structures from this project is a high-end goal for Flora Robotica, and since most of the users at CITA works with fabrication techniques inside the field of architecture on a daily basis.

This brings an important design aspect into question: how can details and optional features be added to the application that can be used to explore the evolution of braided structures in the domain of Flora Robotica, without removing focus from learning the primary user interaction for advancing through the simulation steps. For instance, physical properties could be added as information about the evolved braids, but this might not be interesting for all users, and might camouflage the model making the users unable to evaluate the structure properly. Some users want raw numbers when performing the evaluation step, while others evaluate visually, and by moving around an object. Furthermore, the information presented on the braids might present too much or too little information, and all of the information might not be understandable by all users.

These questions gave raise to the implementation of a menu option which would function as the applications manual for the user to seek help in. The menu would provide details of all the possible operations that could be performed, and could furthermore be extended to adjust the parameters for those particular operations. Likewise, the menu was also thought to contain adjustable settings for the overall application. Settings and flags could then be adjusted from this menu, making it possible to tailor the application to the users needs, including whether braid structures should be represented with raw data and the 3d model, or with the 3d model only.

7.2.2 Providing the EA and user tools for improving the mapping relationship

As seen in the results section, in particular figure 6.3 and figure 6.2, two interesting patterns can somewhat be observed in the user selections. While one strategy can be to constantly keep one genome in the population, another strategy can be to select many genomes. This raises a question in how extra tools can be provided to the user, in order to enhance the communication between the user and the EA. For instance, when selecting multiple braid

structures it could be relevant to mark some of the braid structures as extra important, giving them a higher fitness score than the rest of the population. This in turn can potentially make the EA more efficient when trying to create output solutions that matches the users target solution.

Likewise, another observation made during the experiments was the ability to evaluate the braided structures very quickly if positioned at the right place. For the keyboard experiment, a population size of 9 was often too small, and could easily have been increased to the double, even though they might not have been inspected as thoroughly by the user. In interactive braid evolution, users spend most time in the evaluation step, and by allowing the EA to create a wider range of output solutions, the efficiency of the evaluation step can be increased by reducing the amount of generations needed to be made in order to produce the target solution.

7.2.3 Prioritizing information in the interfaces

Even though there exists several advantages by displaying additional information of either the application state or output solutions, it can potentially make the user focus on something else than the actual purpose when using IEC: to learn the mapping relationship and make the EA produce output solutions that can solve a given problem. For this reason, all debugging windows and buttons was removed from the interface, so that the first thing the user encounters is an initial population of braid structures which needs to be selected. In other words, it was prioritized that learning the mapping relationship was more important than learning the applications functionality. However, information of the applications processes should not necessarily be discouraged in order to maintain simplicity and narrow all focus on learning the simulation steps, rather it should biased through the design. A simple way to achieve this, is to add an option controlling the amount of details shown in the interface.

Furthermore these settings might not need to be limited to the information

provided on the application, but could just as well be expanded to having flags for showing evolutionary parameters, visualizing the CPPNs, or displaying more details on the overall application. It is then up to do the designer of the IEC application to prioritize what information is relevant, and what information should be hidden at the the first encounter of the IEC application, and then leave the user to define what information is relevant for there on.

7.2.4 The possibilities of AR in IEC

As shown in both LEAP videos, several challenges occurred from using the LEAP controller as input device with the LEAP interface. This can be further highlighted by comparing the two experiments average amount of braids evolved and average amount of generations performed by the user. While the amount of experiments conducted on the the LEAP experiment is only five, the experiment results is only useful in combination with the two videos and the summary provided in the user sessions Appendix 2.

By choosing touching as the main approach for the interface, several assumptions on the precision and feeling of the LEAP motion controller was made on how swiping, poking and touching would be interpreted in the simulation. For instance, touch screens uses these techniques for navigation, but has the actual touch screen hardware integrated, instead of using a scanner for detection. Users can furthermore feel the actual touching on the screen, where as in the LEAP experiment, users have to rely solely on visual and audio feedback. The assumptions was then, this the visual feedback would be somewhat sufficient for interacting with the interface, but as seen in both videos, this can indeed be very hard to achieve with the current setup. In relation however, both participants in leapvideo1.wmv and leapvideo2.wmv expresses that performing the operations feels very good compared to using a keyboard and mouse.

Overall, it is hard to say whether the main challenges arises from using

the LEAP interface or by using the actual LEAP motion controller hardware. Rather the results should more be seen as a qualitative experiment in which implementing AR can be used to draw inspiration from, and in which users find it particularly enjoying to use AR as a way of providing input in IEC applications. With both the technology of VR and AR rapidly expanding, this holds promising ways of negating human fatigue - if implemented in a intuitive and natural interface, in which users have to provide minimal effort for all three simulation steps. Since AR introduces Novelty to the IEC application, the novelty also creates new possibilities and challenges that needs to be taken into account.

7.3 Comparing the braids

While a list of points is a very simple representation in terms of 3d modeling, this project shows that by creating a biased modeling algorithm and combining it with CPPNs, IEC, and NEAT, it is possible to evolve biased structures with increased complexity. To see some of these structures compared to two of the most complex structures handmade at CITA studio, see figure 7.1. However, it is believed that further complexity can be achieved by adding new local rules to the evolution step, and thus further biasing the structures in relation to the problem domain.

For this project, the complexity and bias was achieved in several ways. In the modeling process, cylinders would be used as the main modeling element between braid segments, while in the evolution step, a rule for determining the radius values would be added, based on whether their parent had branched. Similarly, rules and threshold governing the splitting of the braid structures was integrated, in this case the rule would add n new nodes to the data structure which would also be input to the CPPN.

By choosing this biased approach, the CPPNs has less freedom compared to the endless forms engine [6], where surfaces could be scaled, warped, ex-

tended and deformed. This means the project has been limited in terms of what structures could be achieved, and shows how it could be useful to find a combination of the two approaches.

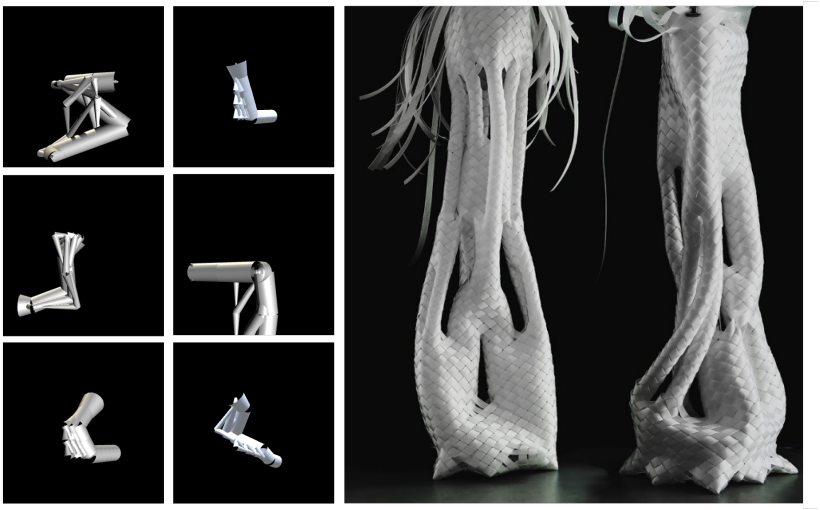


Figure 7.1: A comparison between some of the evolved braided structures from interactive braid evolution and larger and more complex handmade braid structures fabricated at CITA. Right picture is used with permission from CITA studio.

7.3.1 Optimizing the CPPN representation

One of the things lacking in the evolved structures as seen in figure 7.1 is merging the branches back into each other. Adding a rule to merge branches back could be achieved by by evaluating the CPPNs output branching value to be lower than a certain threshold (e.g. 0.2), and check any node neighbors on similar depth would have received a branching value below the threshold as well. If those circumstances were met, a merging of two branches could be created. While the merging process could have several questions on how to achieve it, e.g. do they meet in the middle of the two branches, or should one branch be more offset than the other. Ideally, the merging process should to

some extent be based on some kind of output from the CPPN, in order to make it more varied.

While this adds a new layer of complexity to the CPPN representation and data structure, so too does the freedom in the evolution step for creating complex braid structures, increasing the possibility of increased complexity in braided structures. Likewise, in the current representation, whenever a braid splits, a collection of n - *1* child nodes is added to the branching node, where n is the depth value of the parent node. This holds additional possibilities to be explored, since the amount of added child nodes does not necessarily have to be static, and could likewise be dependent on the CPPN output or based on a different internal rule.

Lastly, the inputs and outputs of the CPPN could receive further experimentation. As described in the background section, some projects include an additional input based on the input values distance from the center [24] [7]. One way to implement something similar was to add the current branching depth of the input node as well. Additional exploration of additional outputs to the CPPN output layer could likewise be added to the representation, e.g. outputs for the the material and radius for each braid segment. Also, since CPPNs can be useful at representing complex patterns, it could be interesting to experiment with CPPN outputs that could create different braiding patterns that could then be applied to the braid structure.

7.3.2 Adding additional layers to the modeling process

Although the results only shows basic skeletons of the braid structures, the approach for having multiple processes allows for easy expanding in each process. One example is to add additional layers of modeling to the modeling process. In figure 7.2 we see a comparison of the same braid when applied with a specific braiding technique (middle image) and with physical relaxation (right object). These two techniques could likewise be integrated to the modeling algorithm, and used in combination with the evolution process.

With each added modeling layer however, it is important to remember that more and more waiting time can occur in the modeling step of the simulation. Currently, the creation and importing of models happens in real-time, making it unnecessary to address human fatigue in the modeling step. However, if new modeling layers were to be integrated to the current modeling algorithm, a balance between the waiting time in the modeling step and the amount of modeling layers added to the modeling algorithm would need to be properly tested. In some cases, human fatigue could be addressed by creating predictor models or fitness functions between each trial as previously mentioned in the background section [18] [17]. This could also be achieved by applying some modeling layers for every *nth* generation,

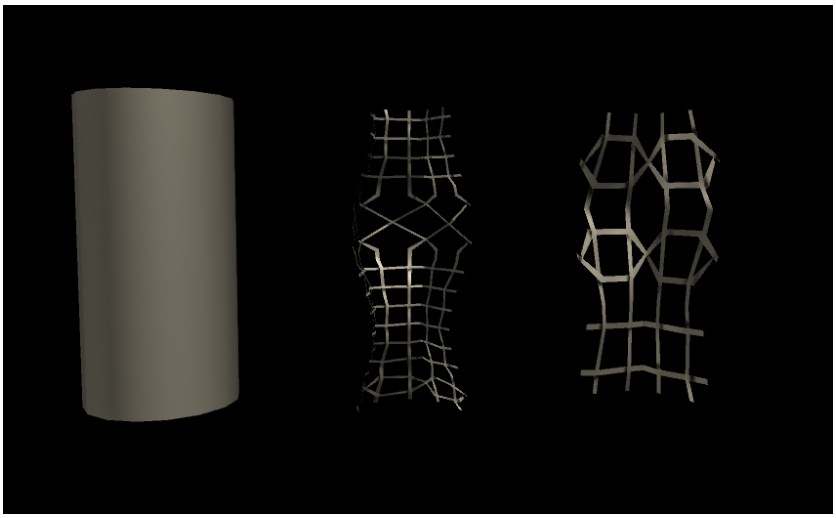


Figure 7.2: A comparison on how different modeling layers can alter and change the shape of the braid structures. The left object is the standard cylinder, the middle object shows a cylinder with braiding applied to it, while the right object has received an relaxation based on its physical properties.

7.4 Extending interactive braid evolution to Flora Robotica

With new technology for allowing novel input devices to be used in IEC, one of the ambitions for interactive braid evolution is to highlight some of the potentials this can hold in areas that focuses on architecture and public installations. While using the LEAP motion scanner holds potential for integrating hand gestures and hand movement to interacting with IEC in a bio-hybrid system similar to the social garden, it is believed that with the new technology emerging from virtual reality and augmented reality, so too does the possibilities of interacting with the social garden in novel ways.

Likewise, the concept of social garden holds interesting potential of influencing the simulation steps, by giving feedback to the user through robotic elements instead of only being limited to the screen. Since one high-end goal for Flora Robotica is the fabrication of braided structures, interactive braid evolution can evolve and send digital representation of braided structures to external processes that can create physical braid structures in the real world, by braiding robots. Integrating robotics into the field of IEC in combination with AR holds novel and interesting possibilities that can be used to overcome some of the main weaknesses of IEC, e.g. human fatigue. In a bio-hybrid system where multiple users may walk around and interact with the garden it would also be relevant to integrate collaborative IEC and likewise encourage certain socially behaviors could be encouraged (similar to Petalz [21]) to further engage users in a meaningful collaboration and together create braided structures.

7.5 Processes and data constraints

In this project, it has been experimented on how simulation and 3d modeling can be achieved in different processes in order to contribute to the overall

functionality of an IEC application. While the typical approach when using multiple processes is to either create and export 3d models beforehand, or create them within the same application, this project illustrates that it is indeed possible to use achieving the same operations at runtime, instead of binding the application to only one process.

This was likewise a prerequisite for the project, since the project engaged in three different fields: interaction design, evolutionary algorithms and architecture. This type of approach can be necessary, when libraries, software, engines or other tools cannot create all of the required functionality for the application. With the right setup, such applications can benefit from a system of connected processes in terms of scalability and flexibility. One of such examples for this project, is the flexibility to add additional modeling layers to the modeling process, to integrate AR and VR into the simulation process, or to integrate a new fabrication process that could create the final braid structures the user had evolved.

However, this also requires a scalable, precise and solid data structure that governs the data being shared between the processes. For this project, the actual representation of the structure that is about to be modeled, simulated or input to the CPPN, should be as simple and precise as possible. Internal process information should be limited to the process itself and only be included in the data packages if necessary. For instance, there is no need to include data of the CPPNs to the modeling process, even though they are critical in the creation of braided structures. Likewise, when sending data between Grasshopper and Unity, it is sufficient to only send the id and file path of the 3d model, even though that this means the data the structure was modeled from is lost.

Another aspect that should be taken into consideration when using multiple processes, is the communication protocol between the processes. This depends on the domain, i.e. a secure connection is preferred if sensitive information is being shared, or a UDP protocol when the reliability of sending

messages might not be of great importance. For this project, setting up a local UDP connection between processes was achieved fairly quickly, while sending and manipulating the data in each process and ensuring the right results are achieved can demand time.

A practical implication of having multiple processes is also is the necessity of setting up the applications processes where the application is executed from. While Unity can create a runnable build of the Unity application, the device the application is being run on needs to have Rhino installed with the required grasshopper plugin, hence a setup and installation of the different processes is required. Similarly a connection needs to be established with a given IP from the current network the computer is using.

8 Conclusion

In this project, two different aspects of interactive evolutionary computation has been explored in the domain of architecture, specifically for creating braided structures that can be inhabited by plants and robotic components. The first aspect of the project would explore the structural possibilities of creating braided structures by using CPPNs combined with the NEAT algorithm. Here, an initial data tree of n would be created, in which each node would be input to the CPPN, followed by the CPPN creating outputs that would create and modify that specific node on the data tree. This CPPN representation was then expanded to allow the CPPN to create branches of new nodes that would be added to the data tree, by adding local rules that would bias the braid structures into a certain category of 3d shapes.

The second aspect would explore how Augmented Reality could improve the efficiency of interactive braid evolution from an interactionist perspective in the field of architecture, which could then provide a small picture on how interactive evolutionary computation could be integrated into other industries. This was achieved by creating two interfaces with different input devices - one with a traditional keyboard and mouse, and the other by using the LEAP motion controller as the AR input. These interfaces and input devices was developed through a user-centered and participatory design process, and was evaluated multiple times during development through user sessions.

The project conducted two experiments exploring both of these aspects by creating storyboards of the user evolved braid structures, recording videos

of users that were using the AR device, and by writing summaries of the user sessions that was carried out during development. These experiments would then provide results that could compare the braid structures that was evolved by the different users and similarly produce results that could be used to compare the challenges and possibilities by using different interfaces.

In general, the project has shown that it is possible to model a CPPN representation that is biased towards a certain category of structures, by adding rules that could expand the data tree in which the braid structures was modeled from. These rules would ensure that variation in the data tree could be created in a flexible and generic way with respect to the domain. On top of this, the modeling algorithm that was used in the modeling process would create cylindrical shapes between each pair of nodes in the data tree in order to further bias the structures towards the handmade prototypes that was created at CITA studio. As seen in the results of this project, the comparison of some of the more complicated braid structures created at CITA studio also highlights that the current CPPN representation needs to integrate a way of the data tree to allow merging of the branches on the braid structures. Still, the storyboards shows that increasingly complex braid structures can be achieved with the current IEC application, and those structures has a reasonable resemblance to the prototypes from CITA in which the goal was to imitate.

This project has also shown the different challenges and possibilities for using familiar input devices versus AR devices, by first creating a framework for analyzing the simulation steps the user encounters for each trial, and then by comparing two different interfaces through user sessions and video recordings. Although AR can provide interesting novelty into the IEC application, so too is the need to find design solutions on how certain operations we perform in our everyday life, can be transferred to IEC. In this project, touch and infrared scanning was used, which could lack precision, partly because the feedback would be limited to the visual and audio channels, and

partly because the scanner could have issues in detecting the fingers of the hand, and making it hard to interact with the ui elements in the simulation. This can be seen in both videos that uses the LEAP motion controller, but it should also be highlighted that the users expresses optimism for the potential of using AR devices for the application, since it "feels much better".

This project has tried to explore how IEC can be integrated into architecture, by using AR and traditional input devices by using different processes from multiple fields in order to create a scalable and flexible application with a solid functionality at runtime. The user evolved structures shows that is indeed possible to create interesting shapes through IEC in the field of architecture, in varied and interesting shapes that fits into a category of certain artifacts that can be inhabited by plants and robotic components. IEC remains an interesting technique in evolutionary algorithms that can function as an multi-disciplinary innovation tool for exploring the possibilities of a problem domain where personal preferences is a part of the solution. IEC also holds relevant possibilities to be extended into other domains in combination with AR, and this project illustrates a simple example on how this can be achieved.

Bibliography

[1] Chadia Abras, Diane Maloney-Krichmar, and Jenny Preece. User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, 37(4):445–456, 2004.

[2] Ergun Akleman, Jianer Chen, Qing Xing, and Jonathan L Gross. Cyclic plain-weaving on polygonal mesh surfaces with extended graph rotation systems. In *Proc. SIGGRAPH*, volume 9, 2009.

[3] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. Evolving competitive car controllers for racing games with neuroevolution. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1179–1186. ACM, 2009.

[4] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. On-line neuroevolution applied to the open racing car simulator. In *2009 IEEE Congress on Evolutionary Computation*, pages 2622–2629. IEEE, 2009.

[5] Maurice Chiodo. An introduction to braid theory. *Msc, University of Melbourne*, 2005.

[6] Jeff Clune, Anthony Chen, and Hod Lipson. Upload any object and evolve it: Injecting complex geometric patterns into cppns for further evolution. In *2013 IEEE Congress on Evolutionary Computation*, pages 3395–3402. IEEE, 2013.

[7] Jeff Clune and Hod Lipson. Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In *Proceedings of the European Conference on Artificial Life*, pages 144–148, 2011.

[8] Maria-Francesca Costabile, D Fogli, Catherine Letondal, P Mussio, and A Piccinno. Domain-expert users and their needs of software development. In *HCI 2003 End User Development Session*, 2003.

[9] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[10] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.

[11] James G Greeno. Gibson's affordances. 1994.

[12] Heiko Hamann, Mostafa Wahby, Thomas Schmickl, Payam Zahadat, Daniel Hofstadler, Kasper Stoy, Sebastian Risi, Andres Faina, Frank Veenstra, Serge Kernbach, et al. Flora robotica-mixed societies of symbiotic robot-plant bio-hybrids. In *Computational Intelligence, 2015 IEEE Symposium Series on*, pages 1102–1109. IEEE, 2015.

[13] Erin J Hastings, Ratan K Guha, and Kenneth O Stanley. Evolving content in the galactic arms race video game. In *2009 IEEE Symposium on Computational Intelligence and Games*, pages 241–248. IEEE, 2009.

[14] Matthew Hausknecht, Piyush Khandelwal, Risto Miikkulainen, and Peter Stone. Hyperneat-ggp: A hyperneat-based atari general game player. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 217–224. ACM, 2012.

[15] Matthew Hausknecht, Joel Lehman, Risto Miikkulainen, and Peter Stone. A neuroevolution approach to general atari game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):355–366, 2014.

[16] Gregory S Hornby, Jordan B Pollack, et al. Body-brain co-evolution using l-systems as a generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 868–875, 2001.

[17] Raffi Kamalian, Eric Yeh, Ying Zhang, Alice M Agogino, and Hideyuki Takagi. Reducing human fatigue in interactive evolutionary computation through fuzzy systems and machine learning systems. In *2006 IEEE International Conference on Fuzzy Systems*, pages 678–684. IEEE, 2006.

[18] Raffi Kamalian, Ying Zhang, Hideyuki Takagi, and Alice M Agogino. Reduced human fatigue interactive evolutionary computation for micromachine design. In *2005 International Conference on Machine Learning and Cybernetics*, volume 9, pages 5666–5671. IEEE, 2005.

[19] Donald A Norman. *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

[20] Jenny Preece, Yvonne Rogers, and Helen Sharp. *Beyond Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[21] Sebastian Risi, Joel Lehman, David B D'Ambrosio, Ryan Hall, and Kenneth O Stanley. Combining search-based procedural content generation and social gaming in the petalz video game. In *Aiide*. Citeseer, 2012.

[22] Sebastian Risi and Julian Togelius. Neuroevolution in games: State of the art and open challenges. 2014.

[23] Elizabeth B-N Sanders. From user-centered to participatory design approaches. *Design and the social sciences: Making connections*, pages 1–8, 2002.

[24] Jimmy Secretan, Nicholas Beato, David B D'Ambrosio, Adelein Rodriguez, Adam Campbell, Jeremiah T Folsom-Kovarik, and Kenneth O Stanley. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, 19(3):373–403, 2011.

[25] Patrikk D. Sørensen, Jeppeh M. Olsen, and Sebastian Risi. Interactive super mario bros evolution. In *Proceedings of the 2016 on Genetic and*

Evolutionary Computation Conference Companion, GECCO '16 Companion, pages 41–42, New York, NY, USA, 2016. ACM.

[26] Kenneth O Stanley. Exploiting regularity without development. In *Proceedings of the AAAI Fall Symposium on Developmental Systems*, page 37. AAAI Press Menlo Park, CA, 2006.

[27] Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007.

[28] Kenneth O Stanley, Bobby D Bryant, and Risto Miikkulainen. Evolving neural network agents in the nero video game. *Proceedings of the IEEE*, pages 182–189, 2005.

[29] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2):185–212, 2009.

[30] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[31] H. Takagi. Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, Sep 2001.

[32] Hideyuki Takagi. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[33] Julian Togelius and Simon M Lucas. Evolving robust and specialized car racing skills. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1187–1194. IEEE, 2006.

[34] Brian G Woolley and Kenneth O Stanley. A novel human-computer collaboration: combining novelty search with interactive evolution. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 233–240. ACM, 2014.

Appendices

A

User session notes:

A.1 Initial interface notes

A.1.1 Petraz feedback: 11/11/2016

feedback: *selects multiple braided structures, understands the simulation steps, has trouble selecting the right braid, noticed slider on the 3rd generation, modeled 63, felt a moderate impact on evolution, would like to walk around the braids.*

own notes: *restart button could be an option when errors occur, viewing and inspecting needs to be improved, braid segments could be visualized.*

A.1.2 David feedback: 11/11/2016

feedback: *paused loop and made evolution process crash, knew how to do selection, noticed slider on 4th generation, thought slider was for adjusting the amount of braids being created each generation, would like to be able to deselect, thinks the mouse movement does not feel good when clicking buttons.*

own notes: *slider seems unnecessary, does not use information from debugging windows, reversing operations should be implemented, evolution loop should have better error handling.*

A.2 Second interface

A.2.1 David feedback: 18/11/2016

feedback: *thinks slider should have a better purpose, thinks the floor on the scene is unnecessary, wants a shortcut for advancing the generation instead of pushing buttons, felt a moderate impact on the evolution, would like to inspect the physical properties.*

notes: pushing buttons moves the camera to an undesired place, the height slider should be removed

A.2.2 Frank feedback: 23/11/2016

feedback: *should have a higher contrast when selecting, accidentially clicked the wrong one, would like the braids to have more complex structure.*

notes: understands the loop, feels a moderate impact on the evolution, thinks the braid should be more varied, recommended L-systems as output value from an ann

A.3 Final interface

A.3.1 Asya feedback: 15/12/2016

feedback: *blows up meshes making Rhino unable to model, does not move around the scene at all, has trouble moving the camera with the mouse, thinks the initial meshes should be simpler.*

own notes: *initial models might have been too branched, branching threshold and values should be re-evaluated, it is hard to inspect the actual structure of the braid since the braid had very thin braid fillaments for this*

experiment, the waiting time was maybe increased too much due to also having integrated the braiding layer in the modeling algorithm.

A.3.2 Yuliya feedback: 12/12/2016

feedback: advances generation even though not all have been imported yet does very quick evaluation and selection, found a proper angle where all braids could be observed at the same time,

own notes: *error handling for advancing generation should be added, fast evaluation and selection steps can be made if camera direction and position is placed correctly.*

A.3.3 Mary feedback: 19/12/2016

feedback: *felt empowered and effective in creating braids, selects more braids than usual, does not move around, is able to evaluate branched structures quickly.*

A.3.4 David feedback: 20/12/2016

feedback: thinks braid variation is good and interesting, thinks braid population should be higher, thinks the scene should be like a museum, crashed unity when saving the files

own notes: *population size could be higher, lack of context in scene might be boring.*

A.4 Initial LEAP

A.4.1 Petraz feedback: 12/12/2016

feedback: *tried to move the cube by carrying with two hands, throwing with one or two hands, and punching has very precise movement and direction of his palms and each finger joint.*

own notes: *touching seems more natural because user receives immediate feedback of the hands position and direction, is able to easily move hands and fingers around the scene.*

A.5 Final LEAP

A.5.1 Petraz feedback: 19/12/2016

feedback: *learned LEAP after around 4 generations chooses several and does not evaluate all braids has a difficult time selecting and swiping the braids made three hands emerge on the screen and crash the process accidentally clapped has hands twice making the generation advance two times in a row.*

own notes: *dragging and clicking sends identical ui events, gestures for selecting middle ui braid could be added for faster selection, too few braids visible, should move braids closer to each other.*

A.5.2 Yuliya feedback: 20/12/2016

feedback: *Learned how to swipe at second generation, thinks selecting a braid is very hard, has a difficult time selecting and swiping, moves hand out to "reset" the current leap state, leap thinks her right hand is her left, thinks clapping is easy.*

own notes: *gestures feels good because it is very easy to perform, swiping feels good but is hard to achieve, selecting is too hard to achieve, and requires a lot of patience.*

A.5.3 Asya feedback: 20/12/2016

feedback: *Thinks swiping with the hand is funny, has trouble selecting, puts her hand through the the ui object, accidentaly advanced generation,.*

own notes: *selection and dragging are reacting to the same events, visual feedback for dragging should have higher contrast, does not evaluate all braids of a generation.*

B

Video recording notes:

B.0.1 LEAP Video 1:

00:30: almost accidentally advance generations
01:30 advances generatio without selecting
02:20 advances generation by accident
02:40 thinks it seelected
03:00 sucessfully selects a braid
03:20 sucessfully selects a braid
03:40 has trouble swiping and selecting
03:50 claps without evaluating all braids
04:25 gets motion sickenss
04:50 application crashes

B.0.2 LEAP Video 2:

00:50 funny to move, learns to swipe quickly
01:00 knows what clapping is supposed to be
01:30 has to show how selection has was done
02:00 open palm for selection,
02:12 right hand gets misinterpreted as the left hand
02:20 gets stuck on the right side of the of the ui canvas
02:30 begins to use slow movements
03:10 tries to stand up
03:15 sits back down
03:50 thinks clapping is a fun gesture
04:40 struggles with swiping and selection once again
04:50 deselects by accident
05:15 thinks swiping is fun when she succeeds