Patrick Anderson (d1108361)

Research and development into an innovative computer game designed to relieve stress.

Abstract

Through this project we look into possible innovative gameplay concepts and how computer programs could be used as a means to reduce stress in people's lives. Presented is research into what work is currently being produced and how it achieves its aims. We discuss our own ideas for innovative game design and analyse our implementation of them into our final product: a small game where the player must speak into a microphone to encourage a virtual plant to grow. The player can also interact with the plant by moving a light source which the plant will attempt to grow towards. The methods used to produce the program and details of how it was written are outlined.

Research

-Game concepts

Computer game design is an area of limitless possibilities. As an entertainment medium I think it has great potential to become far more than what it currently is, as well as having uses outside its traditions. There is a lot of innovative work and experimental games being produced all the time. It is less common in the commercial sector due to the huge investments needed to produce a game today, publishers are less eager to back something which goes away from the tried and tested formats. Innovative games do still get made and can be highly successful. But not in the quantities of the past when far fewer people were required to develop a commercial game. The most interesting and original gameplay can be found now in non-commercial home made games available for distribution via the internet. One game designer who is becoming successful and achieving recognition for his innovative game design is Jenova Chen. Two games of his that I have been looking at are Cloud and fl0w. The first, Cloud is a game he worked with a group of other students on at the school of Cinematic Arts at the University of Southern California. In the game you play as character that can fly through the air and control the clouds in the sky. You can move and collect the "purified" white clouds and use them to convert the grey neutral clouds to white ones. There are also black clouds that can be converted as well. There are nice features such as when the black and white clouds meet, they produce rain. The game is a very simple concept which works well. The challenges presented are to brighten the sky and to produce shapes with the clouds. The whole process of playing the game is very serene and calm. The player gets a sense of freedom from being able to fly as they like with no time constraints. And it is satisfying to pull around huge clouds. The cloud website has this to say about the design process of the game.

"I always wish to make a video game that makes you feel more productive and enjoy your life better. Most video game today is about addiction. But for Cloud, it is designed to be something you can put down and go back to enjoy your life at any time. No failure, no saving. You pick it up and leave it with no second thoughts..."

I think part of the appeal of the gameplay in Chen's games can be compared to the therapeutic nature of performing some repetitive tasks in reality, such as sorting piles of

paperwork. Although without change they can become boring and irritating having your mind occupied on doing something can be very relaxing and produce a sense of achievement when complete. I think the addictive quality of most computer games is important and helps sustain the player's interest in the game. But there is defiantly room for less involved games.

The other game by Chen I looked at: fl0w is an attempt to put into practice Mihaly Csikszentmihalyi's "Flow theory." The player controls an abstract aquatic creature which moves in the mouse direction. The creature can swim and eat other creatures that are swimming around. By eating more and more, the creature can grow and change. There are also other similar creatures you can eat but which can eat you too. The play progresses by moving down to deeper water until the end where once complete you start again in shallow water as a new type of creature. The game has a similar calm feeling to cloud where just moving about is quite satisfying and the ambient sound effects enhance the feeling. But fl0w can become more challenging and pressured as you try to defeat other creatures. If you get eaten by another creature instead of failing the game you move up to shallower water and can move back down to try again. Although the gameplay is continuous it still feels linear to me and that you are just restarting the level.

-Flow

Mihaly Csikszentmihalyi is a psychologist whose work is dedicated to finding out what makes people happy and strives to produce research for a positive effect on people's lives. His theory of flow is about people being at there happiest when they are totally engrossed in doing something. He describes it as follows.

"Being completely involved in an activity for its own sake. The ego falls away. Time flies. Every action, movement, and thought follows inevitably from the previous one, like playing jazz. Your whole being is involved, and you're using your skills to the utmost."

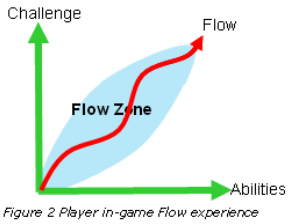


Figure 2 Player in-game Flow experience

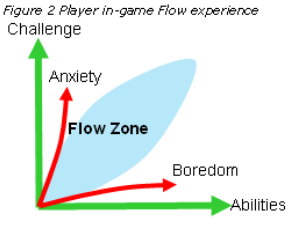


Figure 3 Player encounters psychic entropies

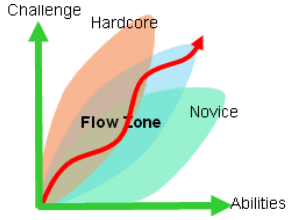


Figure 4 Different players and Flow Zones

Diagram taken from Jenova Chen's thesis "Flow in games."

The concept of flow is like being in the "zone" in a state of total focus and high productivity.

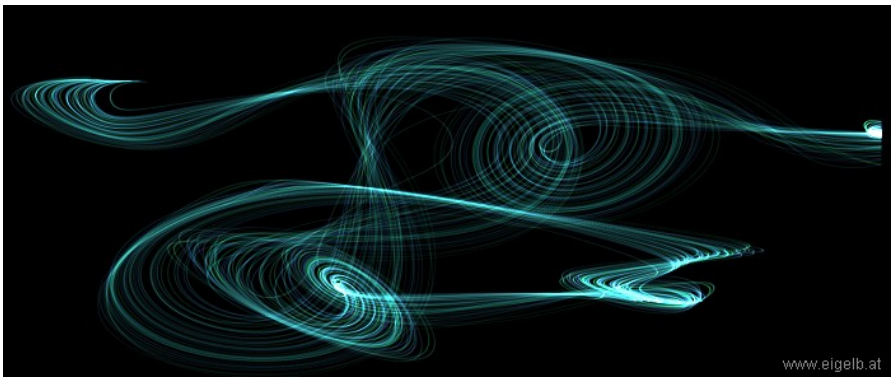
In Chen's thesis he describes his theory of Active Dynamic Difficulty Adjustment which is a system where by a game will adjust its difficulty and challenge corresponding to the player. The aim of this is to maintain the players flow by keeping the gameplay within the safe zone as can be seen in the diagram from his thesis. So the games challenge will keep increasing in alignment with the players abilities. This should then produce the most satisfying and rewarding gameplay.

Adjustable difficulty settings are of course a common feature in almost all games having a hard, normal, easy setting etc. And there are some games that employ dynamic adjustment of difficulty such as commonly RPG enemies may be level dependent. This is more to preserve the mechanics of the game rather than to adjust the challenge to the player. The

first person shooter Unreal Tournament 2004 (by Epic Games and Digital Extremes) features a setting in the player verses computer controlled bot mode which would adjust the AI skill of the bots to keep in alignment with the player's success. So if the player was becoming more successful the accuracy and tactics etc of the bots would increase. This was a great feature and would have served to keep the player in the proposed "flow zone."

Perhaps the dynamic nature of game could be extended beyond the just its difficulty. And have the goals and overall direction of the game be more flexible and based on the player. Some of the most interesting games offer lots of different ways to play. A good example is Freelancer (by Microsoft game studios) which presents a huge universe in which the player can make their own path through and use the many types of gameplay to reach their goals. The game has a continuing story line which keeps pulling the player along but they can move off to do more of what they are enjoying whenever.

What we are interested in looking at is how the concepts of gameplay can be stretched and how they could be used to give the player a new king of experience which relaxes them and is interesting. Maybe what makes a game a game is that it presents a challenge to the player? If it goes too far it becomes something which is interactive but has no set intention. The interactive Java applets of Paul Schmidinger produce nice visual effects when the user interacts with them. This type of thing can be a welcome distraction for a short amount of time for people under stress and can help take their mind off things. They don't present any challenge, unless you try to draw something specific that is. We aim to look at how you could make a game where the aim is to simply relax yourself.



Screen grab from a Paul Schmidinger grappa applet showing the swirling lines which trace out the mouse movements.

-Animating growth

I had already learned a lot about how to produce organic forms with a program. Because it is a common problem and there has been a lot of research into it. Using L-systems is a widely exploited technique for describing growth. There is less research into describing the actual development of the plants shape and its motion as it grows. This is because plants grow so slowly we can't perceive it. For our program we wanted the plant to rapidly grow. We can see how plants grow by making time-lapse films of them. There are a lot of examples of this available on the internet which is done carefully with controlled lighting and basic things like a tripod. But I felt it would be better to observe it myself. Luckily I had planted the seeds from an apple I had eaten (ASDA, golden delicious) and I could watch them as they began to grow. In the warm, unsanitary conditions of our living room they easily germinated.

They grew a lot faster than I expected and I could clearly see the way the leaves and stems spread out. Also they grew strongly towards the sun light and throughout the day followed the sun from east to west. Although plant species grow in many different ways and I am not trying to replicate a specific one it was inspiring to watch one grow closely. And things like the way it breaks out of the pip are small details I was unaware of and would add depth to the program if we implemented them.

Some of the time-lapse images I took of apple seeds growing.

-The use of colour

Colour use is an important consideration in the design of our game further down the line. Because of different colours strong psychological influences. Where colours have effects on people's moods there choice would be important in promoting relaxation. It is known that certain colours are more relaxing and are used for example to paint rooms where this is required. It would make sense to have these be dominant in our program. The most widely accepted relaxing colours are greens and blues. This is because they are the most common in nature being that of the sky and plant life. It is no surprise that our brains favour natural surroundings and using organic forms would be a sensible way to reduce stress. Albeit a recreation on a computer monitor and no substitute for just going outside. That said perhaps one day it will be?

Another possibly interesting use of colour would be to look at its relationship with sound. Both sound and light can both be modelled as waves although their perceivable frequencies being far apart. It has been found that some people posses special abilities whereby they have the gift of perfect pitch and have stated that they can see sound. Or rather specific colours correspond to specific notes and they can see them when they hear the notes. Although most people have no trouble telling apart different colours it is rarer that we can identify specific notes without a point of reference. But perhaps colours could have a relationship with notes to the average person but not as strongly. By maybe trying to find the right match between colours and notes and using those in the program driven by the sound would be the most congruent use.

Idea development

The intention of the game we were prototyping was to experiment with a different take on the normal approach to gameplay. We wanted something that was an interactive experience but did not have the same pass or fail types of challenges in games. My original idea was to make more of an interactive installation where the player would be talked to by the computer and asked to describe what plants and trees looked like as if the

computer had no knowledge of them. Then as they spoke fields of flowers and trees etc would start to generate corresponding to what they said. What the player said would not need to be interpreted in an intelligent way but used to drive parameters in the plants creation.
Within the scope of the project we are limited to how much we can achieve and so only aimed to make a prototype of the type of game experience we envisioned. And hopefully find how much potential it held. I think a main failing of lots of games which have innovative gameplay styles is that they are generally limited and become frustrating for the player after not long. So when compared to fully developed, high budget games people are used to, they don't have a fair chance. Never the less with what we can do, the idea developed to making a test program where a plant will grow and branch. But it has to be encouraged by the player talking or singing to it. In addition to this I wanted to try and simulate the phototropism exhibited in plant growth. So that the player would have the option to move a light source around the plant and have it bend gradually to try to get closer to the source.
Once we had the plant bending towards the light we noticed how much control you had over where the branches were in space. So began to consider other gameplay elements and ideas for extra features we could add such as having areas in space where if you could make the plant reach, they would trigger sound effects or musical notes. The notes could be designed to be relaxing and harmonious or possibly relate to the voice input from the user. For example they could be modified recordings of what was input / is being input. The areas in space could also be used to modify the look of the plant for instance make flowers or leaves appear and change. This kind of addition would hopefully add more of a traditional challenge for the player. They could create unique effects by combining different combinations of areas. But the challenges presented are more personal to the user and they can play it how they want to.

Program Development

I had a clear idea at the start of what the plant system needed to do so I could design how it worked. From what work I had done before on generating trees I knew what to expect as far as the problems I had had and what I thought the failings of my tree program had been. Such as, it not generating continuous geometry along the branches. So I set out to design the system from scratch.

-The branch nodes

The way that the program generated, stored and displayed the plant forms would have to be very flexible to allow for as much diversity as possible. The fact that the plant needed to grow and bend and so animate had to be taken into account.

The solution I decided on was to create a branch node class which when created holds all the information needed about a specific point along the branches of the plant. As well as methods, to manipulate the information and to use it to generate and display the geometry. This way I can create instances of the branch node class dynamically at runtime and link them together with pointers creating a tree data structure with variable numbers of children. This is necessary as I wanted the structure of the tree to not need to be known and to be expandable. With the tree of data created it can then be recursively traversed and any necessary operations can be performed such as displaying the geometry or applying the growth for that frame.

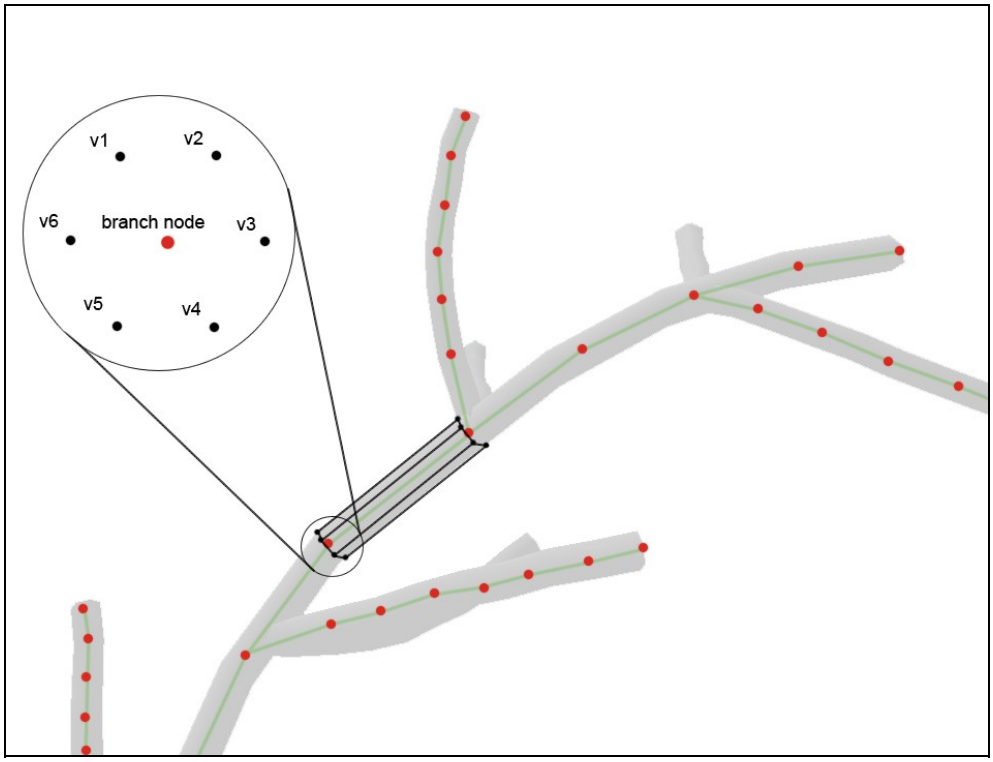


Diagram to show the node structure of the tree.
The red dots show where the nodes are located and the green lines represent their links to one another.

The nodes form a hierarchy system where each node has a link to its parent and a C++ vector list of links to its children, if any. Each node stores its translations relative to its parent therefore in local space. This is a necessity so that branches lower down the hierarchy inherit the transformations of ones further up and the plant can move and grow properly. This presents challenges when trying to work with other objects in world space as I will explain later.

As can be see in the above diagram each node along a branch contains a list of vertices which are stored in an array. These form a ring around the point on the branch and when the tree is traversed are used to draw polygons back to the vertices of the parent node, so forming a tube. The vertices in the ring are stored as world space coordinates as they need to be used to draw back and forth between different nodes.

Each node stores its transformations as rotation about x and z. It is not necessary to rotate y and doing so would cause the geometry to twist. It also stores a length value which is a translation along y. In order to make the branches grow over time the length value can be checked and increased by the grow speed, which is in units per millisecond, times by the number of milliseconds since the last redraw. The branch node also has a maximum length value so the growth is limited.

-Phototropism

To recreate the affect of phototropism on how the plant grows presented another challenge. There were many possible solutions to producing this kind of directed movement. Some of the ones I considered were trying to set-up an inverse kinematics system. Once this was built it would be a case of linking the end effecter to the light source. This technique was originally developed in the field of robotics and is used in

character animation. And whilst it provides easy to direct control of the end of the chain, I don't think it would carry the motion down through the branch and give a natural looking end result. Another possibility was to use 3d Bezier curves in some form to drive the node positions. You could guide the control points by the light and this method would probably produce quite smooth organic shapes. This method seemed to over complicate things for me. I felt that designing something simple based on the simple process a real plant undergoes to direct it's self towards light would be the best solution.

How the branches rotate in the program is a follows. Each node starts with randomized rotations which are kept within reasonable ranges. This produces more natural looking growth and the ranges can be modified to produce more or less contorted forms. Then each node has its own bend limitation which specifies the maximum number of degrees the node can deviate from its original rotations. It is essential to limit the amount the nodes can bend or the branches will just be straight lines towards the light. It also helps to maintain the random shape of the branches.

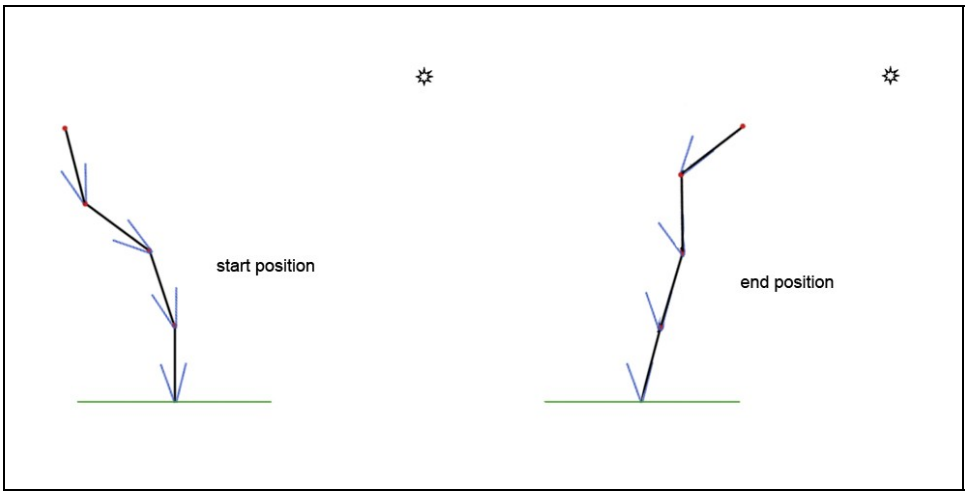
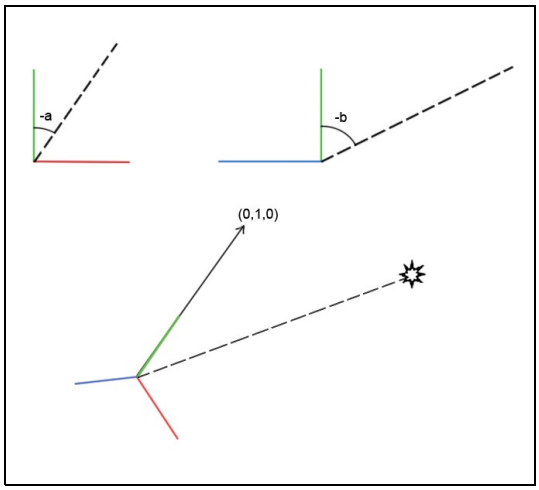


Diagram to show how the branch nodes try to aim towards the light.
The blue lines represent the limitations on the amount the nodes can rotate.

Each frame calculations are performed to get the new rotation values to be set. The amount to rotate by is set by a factor times the number of milliseconds since the last frame as before. The difference in angle between the direction of the branch section and vector towards the light is calculated using the dot product of the two vectors. However the lights position is defined as a world space coordinate and in order to calculate the rotations its location has to be known in the current nodes local space. To get this you have to do a change of basis by multiplying the lights vector by the inverse of the modelview matrix at the current node. When the right values are know the two angles are worked out as 2d rotations. The dot product method only gives the angle of the rotation needed; you also have to determine if it will be positive or negative. A simple test weather the x or the z coordinate of the light vector is negative determines which way to rotate.



The way that the plant grows towards the light worked out well and the way it sways when the source moves looks natural. The speed at which the plant adjusts to the light source can be varied easily and the amount is can bend can be changed. There were of cause issues when developing the movement such as problems with snapping when the branches reached the limits of their rotations but I was able to iron out the problems until I had something I was happy with.

-Splitting

Another aspect of natural growth is splitting along stems and branches. I had been working out the different parts of the program on single stem up until this point. So programming in the ability to dynamically add new growth needed some redesign. I added an attribute to be branch node class which contains the number of splits at that point. To start with each node only has one child or none if it is a leaf node. But as a specified stage of a nodes growth, say when it reaches half its length. If it has any splits defined then new branches can be created and set to be linked to that node. Because the first new node has its parent set to the node which splits, the geometry will be drawn correctly to connect them as well as inheriting the correct transforms. The nodes for the new branch are defined using new so they exist for as long as needed. The memory can be freed by navigating the data tree and collecting pointers to the starts of the branches. Because of the recursive nature the data tree is navigated and because I an using the openGL transformations the program needs to pop and push the modelview matrix to keep in the right location moving about the tree. Because of the limited number of matrices that can be stacked in openGL they have to be used efficiently. By only pushing at split points, popping at leaf nodes and when a whole split node is complete the number required is kept to a minimum. There are still though a finite number of branches a plant could have because of this but generally other limitations will be reached around the same point it seems.

Throughout the program I have used the openGL modelview matrix to perform transformations and stack them. This is not a necessity and the transformations could be kept track of and calculated in my own way. This would be a useful improvement to the program and could yield performance gains. As well as reducing the limitations such as being able to stack more than 32 matrices.

-Roots

I decided to try and implement a root system into the program because I felt it would be an interesting thing to see. As they are not visible they are not usually dealt with in plant generators. Roots grow differently to the rest of plants but there structure is similar so I made them by adapting the branch node system. The root system starts of just like the branch system but growing downwards. Firstly though the roots don't exhibit phototropism they exhibit gravitropism which is growth in the direction of gravity, down into the soil. They do this by regulating the movement of a hormone called auxin in order to cause expansion of specific cell areas and curve the growth downwards. In order to replicate this I simply used the same process of directing growth towards the light source but using a fixed point a long way below the plant instead. The next difference which needed to be corrected was that the stems of the plant bend over time and so sway about as they should but roots have to push through solid ground so cannot sway about like this.

In order to fix this the way they grow was changed, instead of bending over time each node aims as well as it can instantly. Also the nodes grow in length sequentially not all at once. This is illustrated in the diagram below; you can see that each node reaches its maximum length before the next begins.

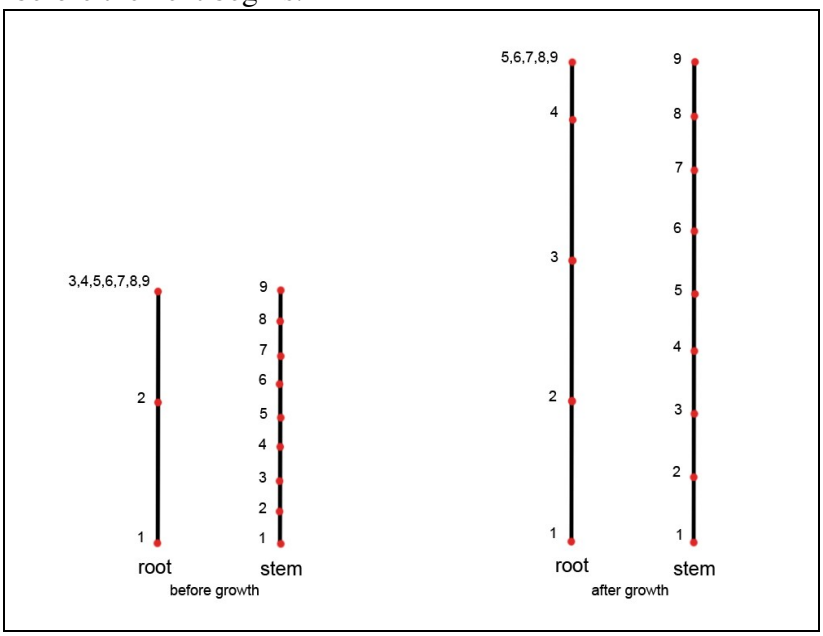


Diagram which shows the difference in how roots are made to grow to stems. The numbered red dots are the constituent nodes.

-Combining the code

Because this was a group project we had to divide up what parts we would work on so Tom focused on the sound and other inputs whilst I worked on the plant system and the overall program. We worked on separate test programs for our different parts. And I gave a copy of the SDL framework I was using to Tom to write into. Then when the parts were ready we just went trough adding the sound and light inputs in and made them drive the parameters. Because we had programmed with what each other was doing in mind the program fitted easily together. I think this was a successful way to work in parallel.

When writing the program it was difficult to keep every aspect in mind when trying to look at the bigger picture. The way that I handled it was rather than trying to make the whole system in one go; I worked out one bit at a time and built it up. So I started just drawing a single branch. Then worked on how to get it to grow, then on making it bend towards a light source etc. This was a good method because when things went wrong I could be assured something was wrong with the current part knowing the others worked. The major downfall was that parts of the code had to be rewritten.

Conclusion

Overall I view this project as a success because we managed to achieve all the things we set out to do and because a lot of the things I tried were new to me I feel I have learned a lot. And importantly by putting into direct practice a lot of the mathematical theory I have learned it has increased and consolidated my understanding. There are definite short comings in the project such as that in its current state I don't think the program would be a hugely effective stress reliever. This was foreseen as it was never

intended to be a completely finished product. But I think that it does show the potential for something which with further development would work. The program functions well and again could be improved further to have the ability to produce far more diverse plants. And if time was put into improving the "looks" of the game making the plant appear more realistic perhaps and adding leaves and flowers would help make for a more polished game. At the time of writing this we are working on producing a version of the game where the player moves the light around and discovers "pockets" of sounds which when found will play. The sound from these causes the plant to grow. This would provide a different type of gameplay to using the microphone and lets the user also create music at the same time. Using a microphone is possibly not to everyone's taste and people may feel self conscious while talking in a room on their own! So this change may make the game more accessible.

Bibliography & further reading

- VARIOUS, Website of interesting time-lapse stock footage, Available from: http://www.naturefootage.com/stock_video/plant_time_lapse_footage.htm [Accessed 13 March 2007].

- CSIKSZENTMIHALYI, M., 1990, *Flow: The Psychology of Optimal Experience*. New York: Harper and Row.

- CHAN, J., Flow in games thesis, Available from: http://jenovachen.com/flowingames/thesis.htm [Accessed 13 March 2007].

- VARIOUS, Cloud the game website: http://intihuatani.usc.edu/cloud/ [Accessed 13 March 2007].

- WEN, H., 2007, Go With the fl0w: Jenova Chen on Console Independence, Available from: http://www.gamasutra.com/features/20070122/wen_01.shtml [Accessed 13 March 2007].

- ANONYMOUS, Pitch and Color Recognition, Available from: http://www.mathpages.com/home/kmath578/kmath578.htm [Accessed 13 March 2007].

- SCHMIDINGER, P., 2002, Grappa, Available from: http://www.eigelb.at/?sID=67 [Accessed 13 March 2007].