Elemental Morphose – Abstracts

Description

Elemental Morphose is a third-person platform/puzzle game for an audience of 10-12 years old. The main goal of the game is to collect all the tokens placed across the level doing so will activate the exit of the level so the player can leave the level.

In order to be able to collect all the tokens in the level, the player needs to solve small 'puzzles' he meets on his way when moving through the level. These puzzles can be solved by transforming (morphing) your character into one of the four elements of nature. Those elements are fire, water, wind and earth.
An example of one of these 'puzzles' can be:
The player encounters a gate he needs to open if he wants to reach the level's exit. For doing so he must first cross a river to then activate a windmill which is connected to the gate. He can turn into water to move through the river and turn afterwards into wind to activate the windmill which will open the gate.

When the players first enters a level he will be in its neutral form. In this state his interaction with the environment is very limited. He must morph into other elements to be able to solve the puzzles.

To morph into one of the four elements of nature the player must find a source for one of the elements and touch it. For example if the player wants to morph into fire he needs to find a burning torch and touch it, this will morph him into fire. Other sources can be a wind blower to turn into wind, a fountain to turn into water or gritstone to turn into stone.

Once the player has collected al the tokens in the current level, the exit of the level will be activated. Once the player enters the exit he is teleported to the next level and will start again in its neutral form.

Development

Team

Our team exists out of four members, we are all students at the Digital Arts & Entertainments course at the University College of West Flanders, Belgium.
There are 2 branches at our course a more programming related branch and a more 3D animation related branch.

Authors: Jyrki Coertjens,
Sander Vanstaen

Our team-members are:

- Jyrki Coertjens: Team leader / Programmer
- Sander Vanstaen: Programmer/Music Artist
- Siegfried Croes: 3D Artist
- Joke van Oijen: Programmer

The 3D-artist made all the models and textures and the three programmers learned how to use the Unreal Development Kit and how to implement all the models & textures the artist made. The level design and animations were also part of the tasks that the programmers had to do.

At the time we started this project most of the team members hadn't met each other before and none of them had worked together at a project before.

Time

Elemental Morphose was a school-project where we needed to prototype a game concept in a time-span of 6 weeks. At the end of these 6 weeks we had to do a presentation about our games which also included a working demo and a game trailer.

During these 6 weeks we had to:

- Work out a game idea.
- Learn to work with the Unreal Development Kit.
- Create a working game prototype and a teaser-trailer.
- Do a couple more other school-projects with tight deadlines which couldn't be ignored.

Actual time spent on Elemental Morphose must be around three weeks. Because of other projects some of the weeks weren't as fruitful as others, so actually working time can be interpreted as follows:

Week 1
First we created 4 game concepts (one for each member of the team), these concepts were afterwards evaluated by our team-members and teachers. Once the final game concept was picked we worked the idea further out and made structural decisions as: who will do what, which game engine will we use, what look do we want for the game, etc.

The rest of the first week was further spent on creating game content, this were especially character models. Other tasks were: understanding how UDK works, creating the level & setting up the game with a third person gameplay.

Week 2
The second week we merged everything from the first week together. We implemented the ability to morph the player's character and added a first form of gameplay.

This meant that the player was able to walk around, morph into some of the elements and activate certain events.
Further assets were also created, so we could start experiment with the gameplay and check what would work and what won't.

Week 3
The third week we finished the gameplay, created advanced materials, added ambient sound & sound effects, particle effects, cutscenes and a basic menu.

The gameplay had some bugs like: trying to activate an event while being in the wrong form from freezes the game or morphing in a certain order disable/enable certain events on the wrong moment.

The advanced materials were a nice finishing touch and gave the character a more 'living' look (before the characters were rather static). Now we were able to add some wavering or flickering motion into the textures. This was especially useful for the fire and wind forms.

Environment

For Elemental Morphose we decided to use the Unreal Development Kit. This is a free version of Epic Games' Unreal 3 Engine, which is updated monthly and is used by both hobbyists as professionals.

The reason for using the Unreal Development kit was because of the short time-span. It was much easier (and therefore faster) to create an environment and gameplay mechanics in a game engine like UDK than building everything from scratch by using for example C++, DirectX & Nvidia PhysX.

The mechanics in Elemental Morphose are done in UnrealScript (a programming language similar to C++) and Kismet. Kismet is a node based programming environment which allows the user to quickly implement interaction, animation, cut-scenes and more in his game.

None of the Intel tools were used.

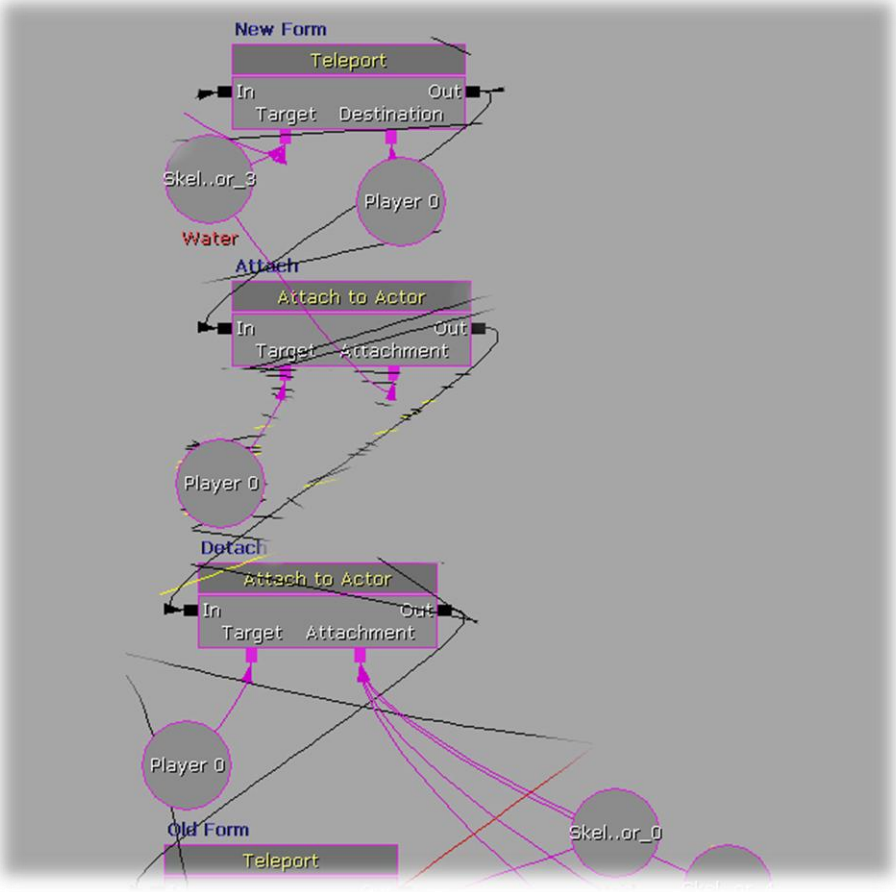
UnrealScript
UnrealScript is an object based programming language similar to C++ and Java. It is also developed by Epic Games.
UnrealScript can be recognized by their .uc extension.

Typical UnrealScipt elements are:

- Each class needs to be ‘expanded’ (derived) from a parent class
- It supports operator, but not method overloading (an exception being when you want optional parameters)
- Each class is part of a ‘package’, this is a collection of different objects who can be only accessed by an actor of the same package (this has consequence that there are no system-wide global functions and variables)
- Variables can be turned into editable properties which can be accessed in the Unreal Editor (UDK)
- The ‘state’ keyword is a script that can group functions, variables and code so it can be accessed by an actor that is in that state.
- UnrealScript is case-insensitive (EXAMPLE == example)

Kismet

As stated above Kismet is a node based programming environment. It allows the user to easily interact with the game actors (i.e.: meshes, lights, materials, camera, pawns, etc.). It also gives the user access to console commands and allows the user to create local variables for the current level or scene.



(Kismet example)

Matinee

Matinee is a part of kismet and it allows the user to easily create animation, cinematics and cut-scenes. Because it is part of kismet you can create in no-time a good looking series of events in your game and give your game that little more depth.

Cascade
Cascade is the particle editor of the Unreal Development Kit. It is easy to learn and allows the user to create any effect he wants to. It is also easy to implement your particles in Kismet and Matinee, which helps in creating a entertaining gaming experience.

Mechanics

All of the gameplay mechanics were done in Kismet. The camera and character movement were done in UnrealScript.

The largest changes we did in UnrealScript were changing the player movement speed and disabling his original jumping ability. We also disabled the loading of any character-models (more information about this later).
Then we also created a small child-class for the original Camera.uc class so we could create the third person view and the free movement camera.

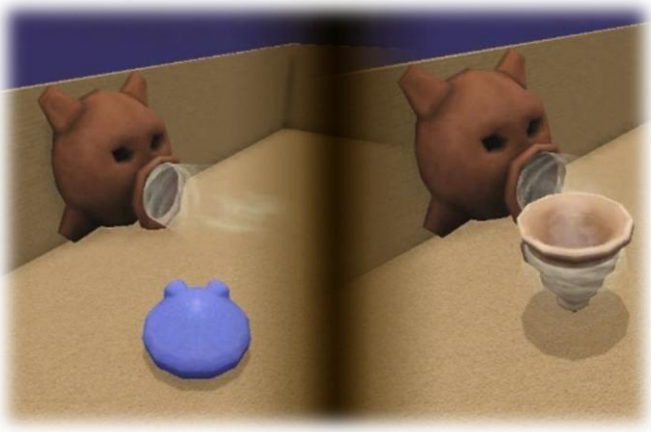
The whole morph and puzzle gameplay is done in one large Kismet sequence. When one of the 'sources' is hit a chain reaction of Booleans is triggered and as final step of the chain the character shape is changed. The Booleans are needed to check at certain circumstances if the player can move through, burn down, activate, etc. some of the level objects.

Morphing of the character is done as followed:
To start with, the Pawn class from UDK has been adjusted so there aren't any character-models loaded. This has the effect that the camera is locked on an empty space where the character used to be, but the player is still able to move through the level.

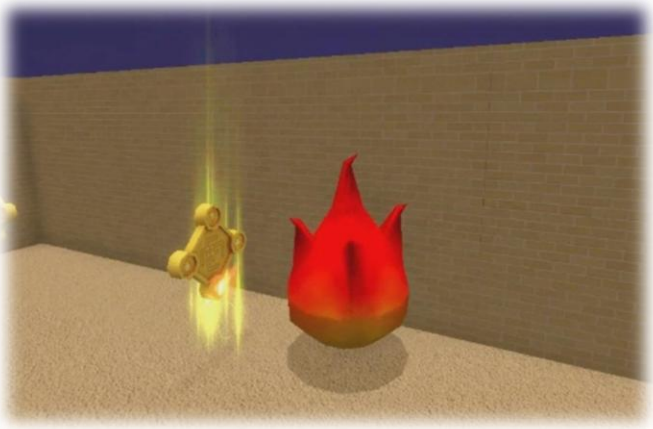
When the level is loaded the five forms of the player (fire, wind, earth, water & neutral form) are already in the level (they are hidden out sight of the player).
What first happens when the game starts is that the neutral form is translated to the position of the player and its orientation is adjusted to the orientation of the player. Then the neutral form is attached to the 'invisible' player. What happens now if the player moves around is that the form move along with the player. Also, when the player turn his character around the form turns along with him. Now it looks like the player is actually controlling the form instead of just an empty spot.

What happens next when the player touches one of the 'sources' is that the current form is detached from the player and moved away. Then the new form is translated

back to the player, orientated to match the previous form orientation and attached to player.
Whenever a player touches one of the ‘sources’ again this sequence is repeated.



Collecting tokens has some similarities:
When a player touches a token a particle system is moved to its position. Then the particle system is activated and the token is removed.
Every time a particle is activated an integer in Kismet is raised with one. Every time this happens, we check if a certain number is reached. If so, a Boolean is set to true, which will activate the exit of the level, so the player can leave the current level.



Content

Almost everything in the game was created by our team, being it either 3D models, particle effects or music.

What follows next is a quick overview of which member of the team made or did what:

- Jyrki Coertjens: morphing the player’s character & gameplay sequences, advanced UDK materials, particles effects, cut-scenes & in-game animations, sound effects, begin menu, character control

- Sander Vanstaen: in-game music, begin menu camera animation, in-game animations, camera-control, teaser-trailer
- Siegfried Croes: models, textures & game concept
- Joke van Oijen: token collection sequence, ingame interface, begin menu

The 3D models were created with 3ds Max 2011 and the textures were created with Photoshop.

Sound was played on a keyboard and recorded with Windows Audio Recorder. Afterwards it was edited with Audacity, so it we could loop it in-game without any interferences.

Challenges

The hardest part was morphing the character into one of the four elements. As stated above we did this by attaching objects to an invisible player. First we tried loading one player model that existed of the 5 forms stacked above each other (with a large space between them). Every time the player morphed the model was supposed to be moved higher or lower. The problem was that UDK didn't allow such real-time model adjustments, so we had to cancel this idea.

Another idea was loading multiple characters with their own forms and transferring control from one character to the other. This could've worked if there was more time to fine-tune it. At the moment of development we didn't get the transfer to work correctly, so this idea was also thrown out of the window.

Another problem was working with multiple users on one level. It is not so easy in UDK to merge Kismet sequences from one file to another. It is possible to export Kismet sequences out of UDK, but importing them may cause some problems. Even if both files contain the same objects, triggers, etc., you need to manually reassign everything in the Kismet sequence all over again (even if the naming of objects is the same). This was a large drawback and forced us to constantly watch over the other team-members' shoulder when merging different things together, so we were absolutely sure that everything worked as it was supposed to.

Future

At the moment we want to keep Elemental Morphose as a prototype example. It is a good showcase for our skills and teamwork, especially because of the very short time in which the game was realized.

Maybe that in the future a new, more finished version of the game will be developed, but if we are going to do this we will move a lot of the mechanics from Kismet to UnrealScript.

Authors: Jyrki Coertjens,
Sander Vanstaen

A couple of reasons for doing so would be:

- Cleaner code instead of a Kismet ‘maze’
- Better performance (since having multiple Kismet sequences running at the same time can cause some framedrop)
- Reusability of code (no need to merge Kismet sequences)
- More customizable code
- Easier merging of content (without the usually reassignment, as which is the case with Kismet sequences)
- Possibility of inserting character animation and soft bodies and by doing so bringing the game to a higher level
- Easier to create multiple levels without having to copy kismet sequences for each level

Probably we will also implement some larger changes to the gameplay and create a more interactive environment. At the moment it’s all fairly static and straight-forward. We’ve been thinking at moving sources so the player so the player needs to intercept them (or evade them) to reach a certain point in the level as the element he intended.

Even some enemies could make their appearance. They could be varying from simple ‘obstacles’ you need to evade or destroy by countering them with the right element. But some larger and tougher enemies could show up. The player needs to manipulate his environment then to overcome this kind of enemy and trap them.

Conclusion

If we look back now at the time when we made Elemental Morphose it was a great and fun experience. We learned much back then, both on a game development level as well on a social level. It was a great opportunity to find out how much could be accomplished in a very short time with just the right preparations, work-willing people and enthusiasm.
We hope can harvest the results of that short experience for a long time, being it either directly or indirectly.

Authors: Jyrki Coertjens,
Sander Vanstaen