ABSTRACT

This paper brings a proposal and general specification of a new project that aims to apply Conceptual Blending (Fauconnier and Turner 1998) as a process of creature generation in games. The project we describe is based on Divago (Pereira and Cardoso, 2003) and should be able to bring novelty among creatures that populate a game. Moreover, it is expected to extend these capabilities to other objects in a game, towards a very dynamic environment.

Since this is a work in progress, we provide the first preliminary results or experimental examples and try to provide the reader with a clear image of the framework to be followed.

INTRODUCTION

Following the ever growing need for novelty in games as well as a growing development community that works in the creation and enhancement of new Artificial Intelligence techniques for games, our work is aiming at the building of a computer-based creativity system that can generate creatures, scenarios, objects and other kinds of game concepts, based on a computational model of Conceptual Blending (Fauconnier and Turner 1998). The generated concepts, although based on previous concepts, should present a newly created structure of their own. This work follows the steps of Divago (Pereira and Cardoso, 2003), a computational model that uses Conceptual Blending (CB). CB is proposed as an explanation for various cognitive phenomena, describing the integration of knowledge from different sources, using a set of cognitive principles and processes. A conceptual blend, or blend, is itself a concept that can achieve independent identity and structure, but at the same time remain linked to its original knowledge (the input concepts). Divago, a computational model for creativity, uses the same principles and processes of CB, resulting in the generation of new concepts. Based on the results achieved by Divago and on the work already done in our project, we have good expectations of ending up with a creative system that will enhance the novelty factor in game content.

KEYWORDS

Creature Generation, Creativity, Conceptual Blending.

CONCEPTUAL BLENDING AND DIVAGO

Conceptual Blending, as proposed by Fauconnier and Turner, may be a plausible model for creativity and it also includes a detailed and explored set of principles and processes, which provides to some extent the construction of a computational model. We will now give an overview of the framework, but to understand it with more detail, we suggest the reader to look in the reference works (Fauconnier and Turner 1998; Fauconnier and Turner 2002).

An Introduction To Conceptual Blending

As stated before, a blend is a concept or web of concepts that is based on previous knowledge, but maintains an identity and structure of its own. The inputs constitute the previous knowledge of a blend, i.e. the concepts that are blended in order to generate a new one. To give an example of a blend, we can think of a mythic character, the werewolf, half man, half wolf, or we can think of computer-related terms, such as “windows explorer” or “recycle bin”.

Starting with the main element of Conceptual Blending, we have the mental space. Mental spaces are knowledge structures regarding a domain, a reasoning, an object or an action. We can blend two or more mental spaces by finding a mapping between them. These mappings, known as cross-space mappings, connect an element of one mental space to another element in the other mental space, which makes these elements counterparts. These elements don’t have to have counterparts, but they can also have more than one counterpart.

Another important part of CB is the frame. When “elements and relations are organized as a package that we already know about, we say that the mental space is framed and we call that organization a frame” (Fauconnier and Turner
There are many types of frames and they can have different degrees of abstraction and functional aspects. For example, we could have “illumination device”, a very specific frame that defines the conditions and concepts that need to be present in a given mental space to be considered an “illumination device”. A light bulb would fit this frame, whereas a computer mouse wouldn’t.

A blend is generally achieved with three steps. The first step involves projecting the elements of each input into the blended space. The second step consists of filling patterns in the blend, when its structure matches information contained in long-term memory. The third step consists of the “simulated mental performance of the event in the blend, which we may continue indefinitely” (Grady et al. 1999).

The projection of data to the blend is a process that involves selection, and is guided by the use of a set of principles that ensure the consistency of the blend. These are known as Optimality Principles or Optimality Constraints.

Due to the lack of space and scope of this document, we advise interested readers to obtain more information about CB in the cited sources, whereas now we’ll continue giving a brief description of Divago.

The **Divago System**

Divago started as a project to model computational creativity, inspired in human cognitive theories, such as J. P. Guilford’s Divergent Thinking (Guilford 1967), or metaphorical reasoning (Leite et al. 2000), which recently led to modelling Conceptual Blending, as it provides the set of principles and processes needed. As far as we know, Divago, contains the only implemented computational account of Conceptual Blending (Pereira 2003).

Divago works with two input domains and a generic space domain, a domain being the equivalent to CB mental spaces. Each of these domains is contained in the system’s knowledge base. It starts by mapping the concepts from the two input domains, also using knowledge from the generic space domain. With mappings done between the elements of each domain, the system tries to create a fourth domain, which is called the blend. This is done with a parallel search engine, a genetic algorithm that parses the large search space of possible projections from the inputs to the blend. To establish a means of evaluation during this process, there is a constraints module (Pereira and Cardoso 2003), based on Fauconnier and Turner’s Optimality Principles. In the end, there is another module that applies knowledge from the generic space, usually a set of rules that will probably alter the blend’s content and structure.

The following figure is a representation of a possible blend between a bottle and a cup. With the help of the Generic Space, the system makes two mappings: the first between “plastic” and “glass” (materials), and the second between “bottle” and “cup” (objects). The dashed lines represent the resulting projections, and the final blend is a “blue plastic cup with a round shape”.

![Figure 1: Bottle and Cup Blend](image)

Divago was recently tested against a set of classic blend examples, taken from literature (mostly Fauconnier and Turner’s). These experiments had the purpose of testing Divago’s ability to model Conceptual Blending. From the results, it is able to find the targets (or very similar solutions) also found in literature. Achieving such, it is our opinion that it reached a capacity for making Conceptual Blending, although still at a relatively basic level in comparison to our own cognition; it is expectable that the system reaches higher levels of complexity after improvements. These and other conclusions will be published in a forthcoming article. These results lead us to think that there may be an opportunity to explore new paths in computer creativity with CB. Being Divago such a generalist system that works with almost every kind of concept constitutes a considerable challenge to its inner workings, because it deals with large amounts of knowledge and data. However, our creature generation system only has to deal with expectable inputs, considerable less amounts of knowledge, and a much more specific blender module. These are the reasons that made us think we could attain slightly better results by restricting the system’s use to a given game environment. This can also result in performance improvements for the system, a determinant factor for the use of this project in computer games, which require real-time processing most of the time.

**CREATURE GENERATION SYSTEM**

Our project is based on the same foundations of Divago. Being Conceptual Blending a promising path for computational creativity, we learned from the internal
workings and results of the generic Divago system, and tried to adapt this information in the building of a more specific system. Although we plan to extend the system capabilities to the generation of scenarios and objects, we are now experimenting solely with the creation of new creatures for game environments.

The system is made up of four main modules, specified in the following figure:

![System Architecture](image)

**Input Module**

The Input Module is responsible for getting and parsing scripts containing two input creatures in the form of concept maps. These concept maps are semantic networks containing a description of the creatures using a set of relations that can be easily reduced to standard attribute-value pairs. For example, a script describing a human fighter could be:

```
is_a(fighter, human)
strength(fighter, 5)
defense(fighter, 4)
food_consumption(fighter, 4)
team_colour(fighter, white)
colour(fighter, flesh, colour)
made_of(fighter, flesh)
pw(head, fighter)
pw(left_arm, fighter)
pw(right_arm, fighter)
pw(torso, fighter)
pw(left_leg, fighter)
pw(right_leg, fighter)
```

We need to make a remark regarding the `pw` keyword, which represents a part-whole relation, meaning, for example, that the head is a part of the whole, in this case, the fighter. All the other relations should be self explanatory. Although the example could be more detailed, this should give the reader a clear view of what is intended as an input script. However, given that we are working on a game system, the contents of a script can vary greatly in detail, depending on the kind of game.

Assuming we have two scripts like the one in the example just given, we can pass them to the input module. Optionally, we can add some restrictions that we want to apply to the blend (e.g., we want the blended creature to keep the input 1 team colour), which are passed into the input module via a third script, normally consisting of what we call goal frames. After the input phase, the module starts to process the scripts, parsing them, looking for syntax errors, and creating internal structures with the input data. If no errors are found, these structures are passed into the next module, the Creature Builder (or blender).

**Creature Builder & Game Knowledge Base**

This second module in the system chain is responsible for the blend process itself. As described in Figure 2, this module interacts with another one, the Game Knowledge Base. This latter module can be individually adapted to a game’s requirements and it contains the generic space of the game environment, which includes the optimality constraints, as well as both the data and the rules to be used during the elaboration phase of the blend.

When the Creature Builder module receives the input structures, it starts to build mappings between the concepts of each input. The mapping process can be more or less restrictive. For example, it can allow a mapping between a head and a leg or it can only allow a head to be mapped with a head. Although the first possibility greatly increases the search space for the blend, that doesn’t mean the results will be more “creative”, because there are a set of constraints that will ensure a desirable coherency in the results, according to the game.

After the mapping is finished, the module starts to make the projections for the blend. This is the most time consuming part of the process, as it needs to search a usually large number of possibilities. This is done with a parallel search engine that works by seeking the projections that give the best possible individual. Current implementation is based on a genetic algorithm, which allows a massive search in a large search space. This process is accomplished with the help of a weighted set of constraints (an implementation of Fauconnier and Turner’s Optimality Constraints), composing the evaluation function, which, in our context, is known as fitness function. During this search phase, one of the constraints verifies if the blend falls into any of the possible frames.

When the search process ends, we have the best blend found by the mechanism, but before delivering it to the Output Module, there are two more phases comprised in the blender module: elaboration and validation.

The elaboration phase tries to enhance and refine the blend. It uses the Game Knowledge Base to search for further concepts that can be added to the new blend. For example,
if the blend results in a bird, the knowledge base may contain information regarding the fact that a bird has feathers and it can fly. Thus, we can add new features to the blended creature. The elaboration phase also applies a set of rules to the blend. As an example, if the blend represents a human fighter with just one leg, and there’s a rule specifying that “if X is human and X has just one leg, then add a wooden leg to X”, the blended creature will end up with a wooden leg.

The validation phase ensures that the blend doesn’t violate any game rules and that it can be successfully converted to the desired kind of output. In this project we’re working with script and 3D outputs, and when it comes to 3D, a lot of problems can arise in the final object if we leave, for example, an axe and a sword in the same hand. The validation phase was conceived to reduce or eliminate these situations. After this stage, the blend is ready to leave the Creature Builder module.

Output Module

The Output Module is responsible for converting the structure containing the blend to the desired kind of output. Before starting with the description of this last module, it is important to note that the whole system was planned to be used in two distinct scenarios. First, it can be used by both character designers and modellers to help them in the development of new concepts and ideas. Second, it can be used in real-time as part of a complete game, generating new content as the game develops. To demonstrate this second possibility we have plans to develop a full-featured game running around the whole blending concept.

Continuing with the Output Module itself, there are two output modes: script and 3D. The script output mode returns a string with the description of the new creature in a format identical to the one used by the inputs. Since this is a simple format, it turns out to be an easy base for altering and experimenting with the scripts, which can help character designers to come up with new creatures derived from the ones generated by the Creature Builder. This format is also useful for storing new creatures in a database thus allowing their reuse as test inputs and to add new content to the Game Knowledge Base.

The 3D output mode is much more complex and it works with a 3D model database where each one of the models is associated with a creature property. It parses the blended creature concepts and generates a new 3D model using new parts acquired during the blend. Currently, the mounting of the model is accomplished with the use of pre-specified connection points. This process also applies colour and texture changes whenever needed by the result. As an example, a gold-plated armour can end as an old rusty steel vest. All the 3D work is done using the well-known Wavefront OBJ file format, so the resulting creatures can be edited in almost any 3D modelling application. There are also plans to develop filters to other widely used file formats, as well as a SDK that can provide custom output for proprietary formats. This way, it can be used in the early stages of game design or has a vital part in a final product. To exemplify some of the results provided by the Output Module, the following picture shows some of the creatures produced:

![Figure 3: Creatures Produced by the Output Module](image)

The creatures represented in Figure 3 are, indeed, two inputs and a blend generated by the Output Module. In order to generate the 3D models, we first created two scripts describing the two inputs, which represent a werewolf and a horse, respectively. These two scripts were then used as inputs for Divago to process. The resulting blend was another script. Then we entered all of the three scripts (two inputs and one blend) into the Output Module, one by one, and this module, using its internal 3D model database, generated the creatures represented in the figure. It is an interesting fact that from all the possible results that could be given by Divago in these specific situation (i.e. blending a werewolf with a horse), this was the one that best satisfied the evaluation criteria. Thus, the mechanism stabilized after achieving this blended creature. In the example, we applied the following weights to the Optimality Constraints: 40% to Integration (measures how much the structures within the blend are part of frames and how well the many frames may fit together), 50% to Topology (measures the maintenance of neighbourhood relations in the whole concept map of the domain) and 10% to Unpacking (measures to what extent it is possible to reconstruct the connections of elements from the blend to the inputs and vice-versa). In this experiment as well as in many others we have done, we used the following goal frames: “creature(_), frame(_), shape_transfer(X,_), shape_transfer(Y,_), \{X\=Y\}”. This means: “a single creature that unifies the two inputs; this creature must maintain the general structure of only one of the inputs.
(either resemble a horse or a werewolf), but must contain at least two different parts from each of the inputs”. This corresponds to a goal setting that came out of many “free association” experiments we made ourselves (we applied consistently these constraints throughout a set of over 50 hand-made creature generations).

It may be of relevance to state that we are only presenting an example that is a result of the interaction between Divago and the Output Module. We have made many more experiments in these conditions, yielding a large variety of results. However it is not intended in this document to evaluate the results obtained with the help of Divago, although it may give the reader an idea of how our system is going to work when concluded.

Obviously, there are some properties of the new creature that cannot be drawn in 3D, such as strength, defense or food consumption. However, these too are handled by the Output Module and are returned in a formatted structure and should be interpreted by the game environment. There are, of course, better and worse programming practices to handle the output of our system, but that information is out of the scope of this document.

Although we described all the processes and phases involved in the creation of a new creature, we must remember the reader that this is a work in progress and some parts of the system are in the development stage. If we show a more confident tone in some passages of this text, it certainly has to do with the background experience gained with Divago, as we already know how the different parts of the system are, at least, expected to work.

The Input Module is completed as well as 80% of the Output Module. The Creature Builder and the Game Knowledge Base are in the early to middle stages of development.

CONCLUSIONS

As we are in the development stage of our project, we provide no results of the blending engine, yet we are using Divago to provide us with some test results, which are of great help both in the building and testing phases of our system. For example, the Output Module was developed using Divago blends as a testing bed. By now, we can assert that both the Input Module and the Output Module are working as expected and should play an important role in providing a stable environment for testing the performance and evaluate the results of the Creature Builder and Game Knowledge Base modules. Based on the results achieved by Divago, we have good reasons to think that the creature creation system will indeed prove itself a good starting point for the introduction of Conceptual Blending as a computational creativity model in the game AI field, and as a base for exploring not only the conception of new creatures, but also the creation of new scenarios, objects and other types of game content in which novelty is an important factor.

Recent games show that AI plays an active role to the success of computer titles. Games are less deterministic, thus more realistic. However, this also leads us to conclude that there is a whole new world of possibilities to explore in the field. There are endless opportunities still open that can help us achieve better games with less effort.

As far as we know, we can find in Microsoft Impossible Creatures (MIC) and Computer Artworks Evolva the works that can be most similar to ours. However, there are considerable differences both in approach and in general objectives. MIC is a game that generates creatures by composition of parts from different sources (which is also possible in our work), but it does not work at the conceptual level, meaning that in principle it is not possible to blend a creature with an object or a scenario, which is viable in our project, depending on the used frames. As far as we know, creatures in MIC all share the same representation scheme (i.e. the same set of slots, but different values), also a big difference to our work. Evolva is also a game for dynamic generation of creatures, but, as with MIC, the space of possibilities is limited to a number of predefined chromosomes, with which the user is allowed to play. We believe that, for achieving a game genuinely dynamic, we need a world that is as open as possible, even taking the risk of generating less valued outcomes, which also happens many times with Divago.

Although this is an academic work with scarce funding, we want to experiment to which extent we can improve content novelty in games, using recent AI techniques, and in which way these are received by both the game industry and the gaming community.

REFERENCES