1. Introduction

%%%%%%%%%%%%%%%%%%%%%%%%%%

\section{Introduction}

%%%%%%%%%%%%%%%%%%%%%%%%%%

This report refers to a work based on a paper from Lucas Kovar, et. al\ref{motiongraphs}, particularly the path synthesis technique described there. Motion Graphs is a method for creating realistic, controllable motion. In Kovar's work a directed graph is automatically generated, this graph contains pieces of original motion and automatically generated transitions. The authors also presented a general framework for extracting a particular graph walk that meet the user's specifications, and apply this framework to a specific problem with different styles of locomotion and arbitrary paths. \\

Path synthesis relates to this final step in the proposed framework, where a user specifies a path and a graph walk is performed in order to generate the locomotion that best suits the path. The goal is to integrate this specific module into a larger virtual character locomotion system. Ideally the user defined path should be approximated by a spline and then a search in the motion graph should be done in order to find a set of motion capture data that minimizes the error. This error is the sum of the squared differences between the defined path and the path that the animation will produce, for that it is used the arc-length distance of the paths. \\

The OGRE engine will be used as the basis for the entire project. A scene with a model is generated where the user can input the desired path to traverse. Once the program finds the set of motions that respect the rules explained above, it is rendered on the scene

1. Implementation details

The implementation can be divided in some different parts. Each one is a fundamental operation off the whole system. The whole process works as one can see on the following workflow on picture XPTO.



* 1. User path definition

User path definition (step 1 in figure XPTO) is the step where the user defines a simple path which the avatar would ideally travel. The user defined path is a sequence of vectors between points chosen by the user in the application screen (picture XPTO). The user, using the mouse, can define a set of points in a plane. This points will be used in the construction of a line that passes trough them. Ideally this line should be a spline (picture XPTO), but in the current state, our work only supports a straight path between each pair of points.



In picture XPTO, one can see as example, in bold, the user defined path, and in dot line a path that an avatar could in effectively follow.

[ NUNO: falar da implementação das linhas? ]

After the path is defined, the user can press the space key (step 3 in picture XPTO), so the next phase, where the animations are evaluated for error, starts.

* 1. How to use animations from motion graphs

To make an animation that runs along the path defined by the user, a set of animations from motion capture data must be chosen. However, not all animations are suitable to fit in a path defined by the user. This arises the question, how to choose animations that fit the path defined by the user? The answer to this question is to associate to each animation, an error. This error will depend of the path followed by the animation, and it will be great if the animation deviates largely from the user defined path and smaller, if the animation, follows a path that is similar to the one defined by the user.



First a segment from the path defined by the user is selected, the path P. For this segment, before the error can be calculated, the software must load and display the animations, so the path travelled by each one of them can be measured. This operation is done following Ogre’s method for displaying animations. To start, all the animations available for the current avatar are loaded. Then, for each one of this animation, we start by running it (P’ on figure XPTO). After the animation reaches the end, we can start to calculate the error (steps 8, 9 and 9 in figure XPTO). The error can be defined as the distance between the arclength of the point where the animation ends, in the animation path (P’) and the point at the same arclength in the user defined path (P).

We then choose the animation with a smaller error, which is the one that will follow a path similar to the one defined by the user.

After this, the process is repeated for all animations in each segment of the user defined path.

[VER NO ARTIGO E COPIAR]

* + 1. How to effectively calculate the error in ogre
* Para cada animação, calcular o arclengh
* Projectar o arclengh no user path
* Calcular o ponto no limite do arclenght
* Usar a equação parametrica da recta para calcular o ponto
* Calcular a distancia entre o ponto obtido e o ponto onde a animação parou

[descrever em detalhe os pontos no fluxograma]

* 1. How to put stuff in motion (for each segment of the user defined path, we have the corresponding animation with the lowest error)

1. Conclusion