AGraphAR: Biblioteca para Suporte a Grafos em Jogos Interactivos de Realidade Aumentada

AGraphAR: A Library for the Support of Graphs in Augmented Reality Interactive Games

Nuno Oliveira, Paulo Dias Almeida
DEI - Departamento de Engenharia Informática
Universidade do Minho
Braga, Portugal
{nmco51845; diasalmeida.paulo}@gmail.com

Pedro Miguel Moreira
IPVC-Instituto Politécnico de Viana do Castelo, Portugal
LIACC-Lab. de Inteligência Artificial e Ciência de Computadores da Universidade do Porto, Portugal
pmoreira@esgt.ipvc.pt

Resumo — **Os recentes desenvolvimentos tecnológicos nos navegadores *web* (*browsers*) relacionados com a computação gráfica, particularmente a linguagem HTML5 e o contexto WebGL tornam possível ter acesso a aplicações de realidade virtual aumentada diretamente a partir do ambiente *web*. Neste artigo apresenta-se uma biblioteca para a representação e construção interativa de um grafo em realidade aumentada implementada para o ambiente *web*. Existe uma grande quantidade de aplicações, nomeadamente no contexto dos jogos sérios e em particular os de carácter educacional, que podem ser implementadas sobre a estrutura de um grafo e que poderiam beneficiar do apelativa visual proveniente da realidade aumentada. A biblioteca aqui descrita simplifica bastante o desenvolvimento destas aplicações encapsulando as operações de gestão do grafo e ocupando-se dos principais desafios provenientes da vertente de realidade aumentada. Por forma a testar e validar e a biblioteca são apresentadas três atividades interativas, que a incorporam como suporte à sua implementação.**

Abstract — **Recent technological developments in web browsers related to the ability to natively capture and process media from local devices and render 3D computer graphics, in particular due to the evolution of HTML5 and its associated APIs, such as WebGL, make it possible to provide virtual augmented reality applications directly from the web environment. In this paper we present a library for the representation and construction of graph based interactive augmented reality games implemented for the web environment. There is a lot of applications, particularly in the context of serious games and in particular those dealing with education, that internally represents information or concepts in a graph-like structure and that could benefit from the appealing look of augmented reality. The herein described library greatly simplifies the development of these applications by encapsulating operations management of the graph and taking care of the main challenges from the aspect of augmented reality. In order to test and validate the library three interactive activities are described.**

*Palavras-Chave: **Realidade Aumentada; Grafos; HTML5; JavaScript; WebGL***

Keywords - **Augmented Reality, Graphs, HTML5, JavaScript, WebGL**

I. INTRODUÇÃO

Os novos standards, recomendações e APIs (*application programming interface*) associados ao HTML5 [1] integram a captura de media e *streams* provenientes de dispositivos locais [2]. Esta capacidade permite capturar, reproduzir e processar *streams* de vídeo, ou outros media, numa página *web* de forma nativa ao navegador (*browser*). Este vídeos podem ser obtidos diretamente e em tempo-real a partir de sensores de imagem locais à plataforma computacional onde está a ser executado o navegador web (e.g. *webcam*). No entanto, no presente, apenas alguns navegadores suportam essa funcionalidade[3].

A tecnologia WebGL [4], permite efetuar a síntese interativa de imagens a partir de modelos 3D ou gráficos 2D em páginas *web*, usando a aceleração do hardware gráfico disponível, através de código JavaScript, de forma nativa ao navegador *web* , i.e. sem recorrer a qualquer módulo (*plugin*) adicional.

Como resultado do desenvolvimento e adopção destas tecnologias torna-se possível ter acesso a aplicações de realidade virtual aumentada diretamente a partir do ambiente *web*. Este ambiente revela-se muito apetecível para desenvolvimento de jogos pois, devido à elevada proliferação e portabilidade dos navegadores as necessidades de instalação e disseminação da solução estão praticamente resolvidas à partida. É também relevante o facto de ao não necessitarem de plataformas específicas, o esforço de configuração para jogos que tirem partido de mecanismos de colaboração remota é mínimo.

Tendo em conta estas considerações propusemo-nos a desenvolver uma biblioteca em JavaScript que permite gerir e interagir com um grafo em contexto de realidade aumentada. O grafo deve ser visto como uma estrutura genérica que pode representar uma infinidade de modelos. A cada nodo do grafo irá corresponder um marcador e cada nodo vai representar uma entidade do modelo, as arestas do grafo vão ser representadas por conexões entre os nodos.

A necessidade de uma biblioteca com estas características surgiu quando nos propusemos a implementar um jogo com

carácter educativo utilizando realidade aumentada para a construção de palavras compostas organizadas em grafo. Verificámos que existiam uma grande quantidade de outros contextos, na área dos jogos e noutras, onde o relacionamento de conceitos poderiam ser beneficiar da adoção deste tipo de estrutura.

Acreditamos que este tipo de aplicações, principalmente devido a sua componente educativa e consequente foco no público mais novo, beneficia com a interface simples e intuitiva de um grafo e da recompensa visual imediata associada a uma resposta correta.

II. GRAFOS EM REALIDADE AUMENTADA

A estrutura base de um grafo pode ser descrita como um conjunto de nodos podendo alguns destes nodos estar interligados através de segmentos de reta, arestas. Um grafo pode ser direcionado ou não direcionado consoante as arestas possuam informação em relação à direção da ligação entre nodos.

Os grafos são utilizados como base para a resolução de vários problemas da vida real. Algoritmos de caminho mais curto e o cálculo do número necessário de *hops* para fazer *flood* a uma rede de comunicação *peer-to-peer*, são exemplos de dois tipos de problema que são resolvidos recorrendo a grafos. A área matemática que define e estuda as características e as operações sobre grafos chama-se teoria dos grafos.

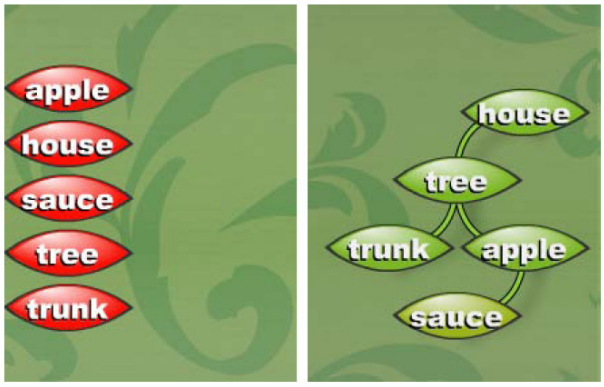

Figura 1. Jogo de criação de palavras compostas (Word Vine). À esquerda encontram-se os nodos disponíveis e à direita o seu relacionamento numa estrutura em grafo.

Uma outra utilização possível para a estrutura em grafo é o relacionamento de conceitos, sendo possível criar mapas mais ou menos complexos de conceitos e dos seus inter-relacionamentos. É sobre esta utilização que se focou o nosso interesse inicial nos grafos, pretendíamos uma estrutura de construção e compreensão simples e intuitiva que nos permitisse implementar aplicações educativas para um público-alvo mais jovem. De forma a aumentar ainda mais a interatividade no processo de visualização e construção do grafo este vai ser representado em realidade aumentada encontrando-se assim o utilizador envolvido no mesmo meio do grafo a ser construído. Na Figura 1 encontra-se uma representação da ideia inicial que nos levou a explorar a representação e construção de grafos em realidade aumentada.

III. ARQUITECTURA DA SOLUÇÃO

De forma a integrar o processo de construção de grafo em realidade aumentada será necessário escolher uma entidade física que seja possível identificar de forma inequívoca, localizando a posição exata no meio físico.

Esta posição vai então ser traduzida para o ambiente virtual onde se simula a estrutura de grafo. Estes elementos físicos vão representar os nodos do grafo e a sua posição vai ser utilizada para fazer a síntese de imagem (*rendering*) de um modelo gráfico que represente o conceito associado ao nodo em questão. Vai ser também preciso definir a operação que representa o estabelecimento de uma conexão entre dois nodos e quais as suas representações no meio físico e no meio virtual.

Para a biblioteca implementada, recorrendo à linguagem JavaScript, começou então por ser definido como representante da entidade de nodo os marcadores de referência. Estes marcadores vão ser identificados e localizados através da biblioteca JS-ARUCO [5]. Os marcadores usados por esta biblioteca estão pré-definidos, ou seja, não e possível definir marcadores personalizados. Um marcador nesta biblioteca pode ser visto como uma grelhas de 7x7 com uma borda preta, sobrando 5x5 células para codificar a informação. Estão disponíveis um total de 1024 marcadores. Cada linha da grelha deverá corresponder a um dos seguintes padrões:

white -black -black -black -black

white -black -white -white -white

black -white -black -black -white

black -white -white -white -black

A interação entre os *web browsers* e a biblioteca é garantida pelo HTML5 e pelo WebGL. O HTML5 para além de providenciar mecanismos que permitem criar uma interface gráfica de grande interoperabilidade entre os utilizadores e as soluções produzidas, permite também aceder a *streams* de vídeo. As fontes de vídeo podem ser de vários tipos, desde uma *webcam* a um vídeo armazenado no computador. O WebGL torna possível usar a aceleração 3D do hardware gráfico a partir do código JavaScript, tornando possível o *rendering* de modelos 3D de grande complexidade. A biblioteca three.js [6], é usada na interação com a tecnologia WebGL. Na Figura 2 encontra-se um esquema das relações entre as várias camadas envolvidas na solução proposta.

IV. IMPLEMENTAÇÃO

O objetivo principal da biblioteca implementada é a disponibilização de uma forma rápida e eficiente de criar e gerir um sistema de realidade virtual aumentada interativo cujo modelo pode ser representado por um grafo. A biblioteca procura ser suficientemente abstrata para permitir que os utilizadores possam criar sistemas que correspondam às suas necessidades.

A biblioteca recebe no seu construtor o *stream* de vídeo onde vai realizar a procura dos marcadores e o ambiente de *rendering* onde os vai representar. Existem duas estruturas de dados base que correspondem aos nodos e arestas do grafo. O utilizador poderá adicionar um novo elemento a cada uma destas estruturas em qualquer altura do *runtime*. A entidade que

representa um nodo tem como atributos a posição no ambiente de *rendering* correspondente ao marcador que identifica este nodo e o modelo gráfico que irá ser utilizado para o representar. São disponibilizadas três funções para a construção de nodos diferenciadas no tipo de modelo gráfico, sendo possível criar um nodo representado por uma *string* de texto 3D ou por um qualquer modelo gráfico fornecido (sendo aceite os formatos .dae e .json).

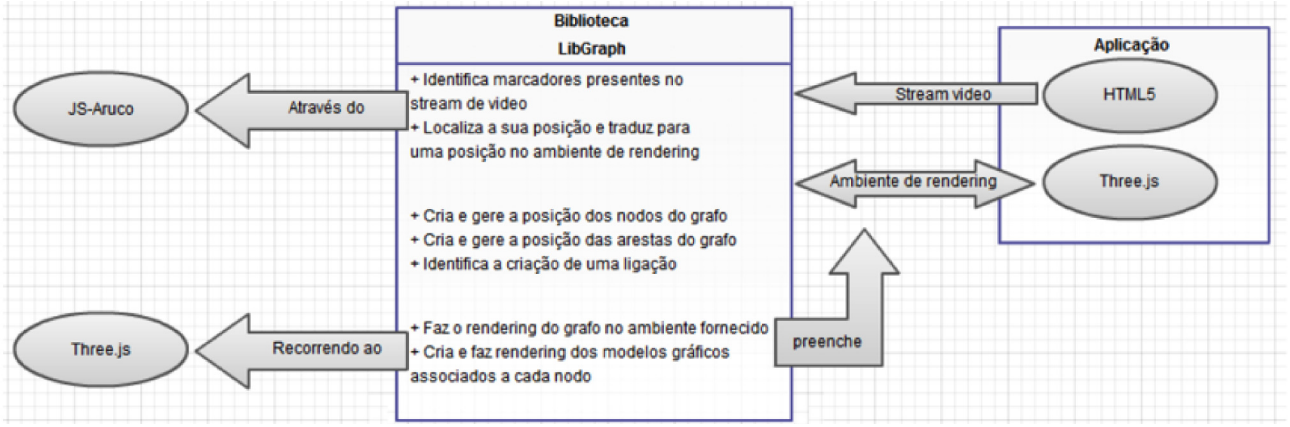

Figura 2. Arquitectura da implemetação proposta. As aplicações terão de fornecer à biblioteca o stream de vídeo onde vão ser localizados os marcadores e o ambiente de *rendering* para preencher com os modelos gráficos. A biblioteca fica assim responsável pela identificação e localização dos marcadores, dados que vão ser utilizados para atualizar a posição dos nodos correspondentes no grafo. Após atualizar a posição dos nodos a operação de estabelecimento de conexões é efetuada e os modelos gráficos associados aos nodos e conexões são introduzidos no ambiente de *rendering*.

O funcionamento base de uma aplicação implementada sobre esta biblioteca corresponde assim à captura de um *snapshot* proveniente do *stream* de vídeo sendo a imagem obtida posteriormente analisada pelo JS-ARUCO de modo a identificar e localizar os marcadores presentes na imagem. Uma vez identificados os marcadores, verifica-se se estes correspondem a algum dos nodos registados, se tal se verificar, a posição do nodo em causa é atualizada e o modelo gráfico associado a este passa a fazer parte da imagem da cena a sintetizar.

Após a atualização da posição dos nodos é ainda tomada a decisão de estabelecimento de conexões com outros nodos. Esta decisão é baseada na distância no espaço 3D verificada entre cada um dos nodos presentes em cena, se esta distância for inferior a um limite definido pelo utilizador é então estabelecida uma conexão ou aresta. As arestas são representadas por uma linha azul ou vermelha a ligar os nodos, identificando as conexões corretas (registadas na estrutura de arestas) e incorretas, respetivamente.

Os principais problemas que foi necessário resolver para a implementação da biblioteca acima definida decorreram da localização dos marcadores de referência, da decisão de estabelecimento de conexão entre nodos do grafo e da integração das várias tecnologias que esta solução recorre.

A localização de marcadores proveniente do JS-ARUCO varia muito entre quadros, sendo bastante dependente de fatores de calibração como iluminação, o tamanho dos marcadores, etc. Esta instabilidade na identificação leva a fenómenos de *flickering*, onde os nodos identificados em cena ficam em constante alteração devido ao facto do JS-ARUCO perder a temporariamente sua posição.

O resultado proveniente deste fenómeno era então muito instável, tanto ao nível visual, onde o modelo gráfico correspondente ao nodo se encontrava sempre a tremer, como ao nível aplicacional, pois de cada vez que se deixava de identificar um nodo este era retirado de cena perdendo-se todas as ligações onde este correspondia a um dos vértices¸ A solução a que chegámos para este fenómeno baseou-se no adiamento da decisão de retirar um nodo de cena. A cada nodo faz-se corresponder um contador que será incrementado sempre que se verificar um quadro em que o nodo não é detetado, quando este contador atingir um valor pré-definido retira-se então o nodo de cena. Os resultados obtidos com esta solução revelam-se muito mais estáveis, introduzindo apenas uma muito pequena latência na retirada de um nodo de cena.

Na definição da operação de estabelecimento de conexão entre dois nodos consideramos duas alternativas, a definição de um marcador pré-definido que exercesse essa função, ou através da distância espacial entre nodos. Após algumas experiências verificamos que a utilização de um marcador de estabelecimento de ligação levava a uma interface mais complicada, tanto em conceito (principalmente para um público mais novo), como ao nível físico, onde se iria adicionar mais um marcador para manusear em conjunto com os vários nodos do grafo. Decidimos assim basear o estabelecimento de ligações na distância espacial entre nodos, este conceito revela-se bastante intuitivo (cada nodo irá ter uma zona de atração onde se estabelecem conexões), simplificando também a interação necessária para construir um grafo.

V. RESULTADOS

Para testar e validar a biblioteca desenvolvida procuramos implementar a aplicação que nos levou a considerar a necessidade desta. Esta aplicação representa um jogo de expansão do vocabulário para utilizadores que estejam a dar os passos iniciais na aprendizagem duma nova linguagem. Cada nível deste jogo introduz um conjunto de palavras que irão ser representadas como nodos. As palavras disponíveis

relacionam-se entre si de forma a originar expressões de palavras compostas. O jogador tem, para cada nodo, estabelecer as conexões que representam todas as composições de palavras possíveis. Na Figura 3 pode-se observar a informação de nodos e arestas definidos na Figura 1.

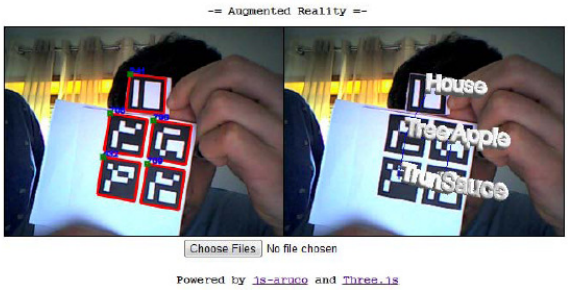

Figura 3. Marcadores impressos na folha encontram-se à distância válida para estabelecer conexão.

Na segunda aplicação exemplo desenvolvida procuramos demostrar a facilidade de utilização de modelos 3D em soluções que se suportem na nossa biblioteca. Todo o processo de carregamento e síntese dos modelos é encapsulado da aplicação tendo esta apenas que indicar qual o caminho para o ficheiro (.dae ou .json) que contém o modelo. Na Figura 4 pode se ver a aplicação implementada, que tem como objetivo categorizar um conjunto de animais em relação à sua classe e ordem.

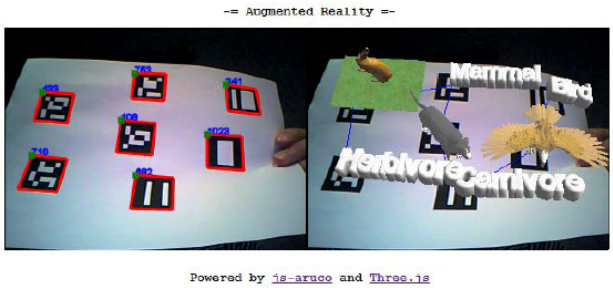

Figura 4: Aplicação de categorização de animais. Todas as operações relacionadas com o carregamento e gestão dos modelos gráficos ficam a cargo da nossa biblioteca

A última aplicação exemplo desenvolvida pretende demonstrar que a utilização da biblioteca não retira nenhuma liberdade ao código aplicacional. As aplicações que recorram à biblioteca continuam a ter acesso aos conteúdos de cada modelo, bem como à estrutura do grafo e propriedades decorrentes dessa estrutura. Esta utilização tem como objetivo organizar um conjunto de letras de maneira a formar uma palavra, quando todas as conexões entre palavras se encontrarem estabelecidas é efetuado o *rendering* da palavra completa. A aplicação verifica assim em cada iteração o estado do grafo gerido pela biblioteca, neste caso particular inquirindo sobre o número de conexões identificadas em cena. Quando o número de conexões chegar ao número necessário para formar a palavra, a aplicação vai percorrer os nodos que definem os vértices dessas conexões, acedendo ao modelo gráfico associado de forma a concatenar as letras que o definem. A *string* que representa as letras concatenadas é então utilizada para construir um modelo gráfico da palavra completa. Uma ilustração desta aplicação é apresentada na Figura 5.

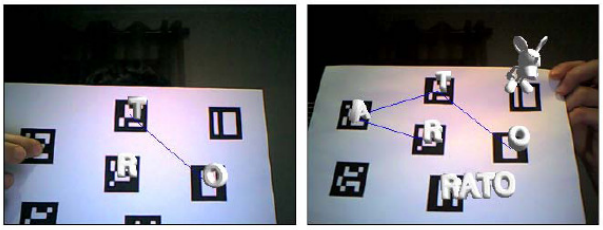

Figura 5: Aplicação de formação de uma palavra. Na figura da esquerda encontra-se tapado o marcador associado à letra A impedindo a formação da palavra. Quando este é destapado a aplicação verifica que o número de conexões é suficiente para formar a palavra e introduz na biblioteca nodos com as letras concatenadas e com uma representação gráfica da palavra.

VI. CONCLUSÃO E TRABALHO FUTURO

Na realização deste trabalho verificamos que o *rendering* de ambientes gráficos interativos na *web* é um meio de desenvolvimento de jogos muito apetecível e em constante crescimento, existindo ainda uma grande quantidade de funcionalidades a explorar. Nomeadamente, os jogos com propósitos educativos direcionadas a um público mais novo podem beneficiar muito desta área, dada a sua facilidade de acesso e apelativo visual proveniente da realidade aumentada. Este trabalho tem um enfoque particular nos jogos de relacionamento de conceitos, normalmente baseados numa estrutura de grafo. Os protótipos produzidos demonstraram a utilidade da biblioteca no desenvolvimento de uma aplicação deste tipo, encarregando-se das preocupações provenientes da utilização de marcadores e dos modelos gráficos necessários, sem retirar liberdade ao código aplicacional para manipular em qualquer momento do *runtime* todas as estruturas do sistema. Os próximos passos a dar na implementação da biblioteca apresentada seriam no sentido de documentar e dar mais robustez ao código desenvolvido, introduzindo também novos operadores e funcionalidades sobre a estrutura de grafo em realidade aumentada.

REFERÊNCIAS

[1] R. Berjon, T. Leithead, E.D.Navara, E.O'Connor, S. Pfeiffer (eds), *HTML5 - A vocabulary and associated APIs for HTML and XHTML, W3C Candidate Recommendation 17 December 2012*, W3C, 2012. Disponível: http://www.w3.org/TR/2012/CR-html5-20121217/ (acedido em 12 de abril de 2013)

[2] D. Burnett; A. Narayanan (eds). *Media Capture and Streams. 28 June 2012. W3C Working Draft*. W3C, 2012. Disponível: http://www.w3.org/TR/2012/WD-mediacapture-streams-20120628/ (acedido em 12 de abril de 2013)

[3] Alex Deveria, Can I use getUserMedia/Stream API? In Compatibility tables for support of HTML5, CSS3, SVG and more in desktop and mobile browsers. 2013. Disponível: http://canisuse.com/stream (acedido em 12 abril de 2013).

[4] C. Marrin (ed). WebGL *specification.Version 1.0.1*. Khronos Group, 2012. Disponível : https://www.khronos.org/registry/webgl/specs/1.0/ (acedido em 12 de abril de 2013)

[5] J. Mellado. js-aruco : A JavaScript library for Augmented Reality Applications, 2012. disponível: https://code.google.com/p/js-aruco/ (acedido 12 de Abril 2013)

[6] R. Cabello, Three.js authors. *three.js : JavaScript 3D library*. 2013. Disponível: http://threejs.org/ (acedido 12 de abril de 2013).