O Ambiente de Programação Visual Mosaicode

Flávio Luiz Schiavoni, José Mauro da Silva Sandy, Thiago Thadeu Souto Cardoso
André Lucas Nascimento Gomes, Frederico Ribeiro Resende

[1] Departamento de Computação (DCOMP)
Universidade Federal de São João del-Rei (UFSJ)
São João del-Rei – Minas Gerais – Brazil
fls@ufsj.edu.br, jmsandy@gmail.com, souto.t@hotmail.com
andgomes95@gmail.com, fredribeiro97@gmail.com

Abstract. *The process of development in Digital Arts involves specific knowledge in arts and programming knowledge for the creation of new tools. However, without the coding ability, this process is hampered. To simplify the creation of programs with artistics purposes, making it simpler, Visual Programming Languages are used. In this article, we'll introduce the Mosaicode visual programming environment that aims to help non-programmers create their own applications by providing the source code generated by the project for possible customizations.*

Resumo. *O processo de desenvolvimento em Arte Digital envolve conhecimento específico em artes com conhecimento de programação para a criação de novas ferramentas. Porém, sem a habilidade de codificação, este processo é dificultado. Para simplificar a criação de programas com propósitos artísticos, tornando-a mais simples, são utilizadas Linguagens de Programação Visual. Neste artigo, será apresentado o ambiente de programação visual Mosaicode que visa auxiliar não programadores a criar suas próprias aplicações, disponibilizando o código fonte gerado pelo projeto para possíveis personalizações.*

Link para o vídeo: https://mosaicode.github.io/#video

1. Introdução

Os processos de criação no campo das Artes Digitais vão além da utilização de ferramentas já existentes e passa pelo desenvolvimento de novas aplicações computacionais para o fazer artístico. Isto faz com que artistas digitais sejam também programadores e possuam um conhecimento técnico computacional mais elevado do que o de usuários comuns. Como lembra Miller Puckette, autor do Pure Data: "O processo de fazer música computacional é, primeiro, escrever um software, e então fazer música com ele"[Puckette 2002][1].

Para permitir o desenvolvimento de aplicações por artistas digitais, surgiram várias linguagens de programação de domínio específico (DSL - *Domain Specific Programming Languages*) para a criação de Arte Digital com ambientes de programação visual e geradores de código que auxiliam no reuso de código e prototipação rápida.

Unindo a simplicidade da programação visual com a abstração de linguagens específicas de domínio, apresentamos o ambiente de Programação Visual Mosaicode. Este ambiente foi criado com o intuito de diminuir a barreira entre artista e a criação e o desenvolvimento de aplicações artísticas sendo voltado para o domínio específico das Artes Digitais.

[1] "*The process of doing computer music is, first, to write software, and then to make music with it.*"

1.1. Histórico

Em 2003, dentro do edital CT-INFO 2003 - Software Livre da FINEP, o grupo de pesquisa multidisciplinar S2i da Universidade Federal de Santa Catarina (UFSC) desenvolveu uma aplicação chamada Harpia[2]. O Harpia é um ambiente gráfico de programação, na concepção de software livre, para auxílio na educação, treinamento, implementação e gerenciamento de sistemas de visão computacional.

Em 2014, como uma ramificação do Harpia, pesquisadores do Departamento de Computação da Universidade Federal de São João del-Rei (UFSJ), iniciaram uma refatoração de código desta ferramenta com o objetivo inicial de reativar a ferramenta que se encontrava inativa devido a dependências descontinuadas. A refatoração iniciou-se com a remoção e o isolamento destas dependências em camadas e componentes específicos para evitar problemas do mesmo tipo no futuro. Na refatoração do código, a ferramenta foi totalmente reescrita e sua arquitetura foi modificada de maneira a trabalhar com extensões e *plugins*. Com o desenvolvimento de novas extensões para o domínio da Arte Digital a ferramenta foi renomeada para Mosaicode e distribuída com a licença *GNU GPL V 3.0*.

2. Descrição e motivação

A Arte Digital representa hoje a convergência entre Arte, Ciência e Tecnologia desafiando artistas a conectar Arte com novas tecnologias [Grau 2003]. Este desafio impactou e transformou radicalmente as atividades artísticas tradicionais como música, pintura, dança e escultura, permitindo o surgimento de formas de arte completamente novas como *net art*, *media art*, instalações digitais e realidade virtual [Wands 2007].

Artistas digitais muitas vezes possuem formação em alguma arte tradicional, como música, cinema ou pintura, e têm uma grande dificuldade de iniciar sua pesquisa e trabalho com arte digital devido ao não conhecimento de algoritmos e lógica de programação tradicional.

Este é o contexto e a motivação de desenvolver um ambiente de programação visual para programação de Arte Digital. Neste ambiente, a programação é feita de maneira visual e diversas funcionalidades deste domínio de aplicação estão agrupadas em blocos que podem ser utilizados para o aprendizado por usuários leigos ou para a rápida prototipação por usuários experientes. Com isto oferecemos uma maneira rápida, simples e intuitiva de artistas digitais desenvolverem obras digitais a serem usados em apresentações artísticas e instalações sem requerer que os mesmos aprendam conceitos de uma linguagem de programação convencional.

3. Principais funcionalidades

O Mosaicode utiliza o paradigma de encapsulamento de código em Blocos onde cada Bloco encapsula uma determinada funcionalidade no ambiente. A ferramenta conta com alguns conjuntos de Blocos com funcionalidades distintas, como apresentado na Figura 1-1, que podem ser combinados por meio de conexões para a criação de aplicações. Estas aplicações são desenhadas na forma de Diagramas, como apresentado na Figura 1-2. Um Diagrama pode ter ainda comentários, que servem para documentar o código. Para a geração de código, um padrão de código é utilizado de forma que a alteração e manipulação deste *template* permita criar diferentes combinações dos mesmos trechos de código.

Blocos: Os Blocos do Mosaicode são as unidades de programação. Os Blocos são separados por linguagens e agrupados por categorias, sendo que estes possuem um conjunto de propriedades estáticas, configuradas na própria ferramenta Mosaicode ou propriedades dinâmicas, configuradas

[2]Página do projeto: `http://s2i.das.ufsc.br/harpia/en/home.html`.

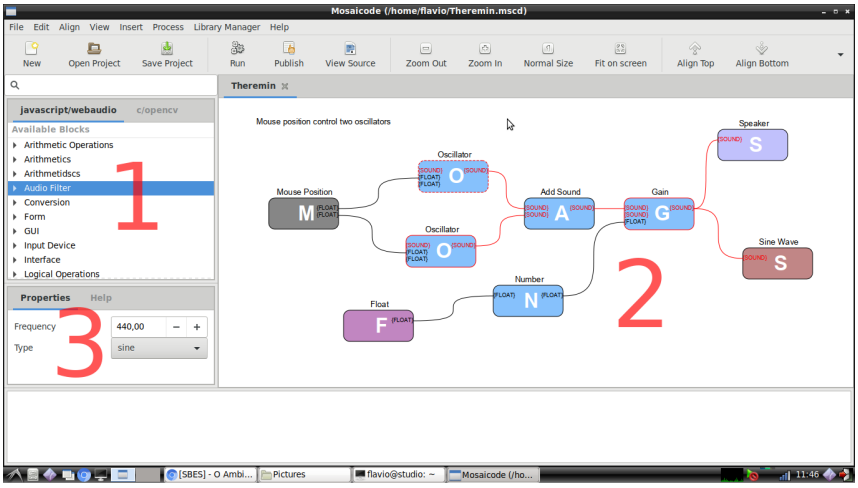

Figura 1. Ambiente de programação Visual da ferramenta Mosaicode.

por outros objetos. A configuração das propriedades estáticas é apresentada na Figura 1-3. As propriedades dinâmicas dos blocos são representadas na forma de portas de conexões. As portas de entrada são utilizadas para configurar dinamicamente um Bloco e as portas de saída permitem utilizar a saída de processamento de um Bloco para a configuração de outro Bloco. Entradas e saídas são combinadas por meio de Conexões.

Conexões: Conexões são as ligações entre os Blocos. Uma conexão possui um tipo e apenas conexões compatíveis podem ser feitas, isto garante que um Bloco só receberá valores que sabe como utilizar. A conexão entre Blocos serve para a troca de mensagens entre os trechos de código de cada Bloco e também para definir a ordem de geração do código. A combinação de Blocos e conexões geram diagramas de programação.

Padrão de código: Blocos e Conexões costumam pertencer a um domínio de aplicação específico e são desenvolvidos para uma linguagem específica e usando uma API desta linguagem. Da mesma forma, um padrão de código específico (*template*) pode ser criado para um determinado domínio e linguagem de programação. Ao Gerador cabe processar individualmente cada bloco e conexão para a criação do código-fonte final do Diagrama baseando-se neste *template*. O *template* define os trechos de código de cada bloco para garantir a correta substituição e concatenação dos mesmos.

O código gerado é compilado ou interpretado, e posteriormente executado dentro do próprio ambiente, o que causa a impressão de que o ambiente executa ou interpreta o diagrama gerado. O ambiente possui ainda diversas implementações que permitem a colaboração em arte digital. Extensões criadas por um usuário podem ser compartilhadas, os diagramas gerados podem ser compartilhados e o código gerado pode ser compartilhado. Tudo isto por meio de um servidor web embutido no próprio ambiente.

4. Arquitetura da ferramenta

A arquitetura atual da ferramenta baseia-se no modelo *MVC*, conforme ilustrado pela Figura 2, e conta com as seguintes camadas:

Camada de Persistência: é responsável por salvar e carregar os Diagramas, Blocos, Portas e *Templates* da ferramenta. A camada de persistência utiliza a camada de Modelo e seus métodos são chamados exclusivamente pela camada de Controle.

Camada de Controle: é responsável pelas ações do ambiente. É nesta camada que ocorrem a ligação entre as demais classes do ambiente, garantindo um baixo acoplamento do código. Todas

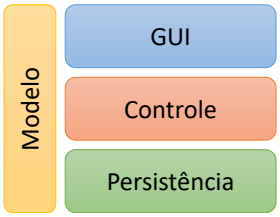

Figura 2. Modelo Arquitetural.

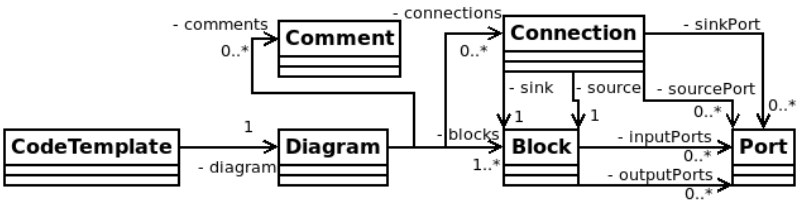

Figura 3. Diagrama de classes da camada Modelo.

as ações do ambiente estão definidas na camada de controle que toma decisões sobre quais classes devem tomar parte de quais ações.

Camada de GUI: define a interface de usuário e é baseada no Gtk 3. Todos os componentes gráficos do ambiente estendem uma classe Gtk e especializa esta classe para as necessidades da ferramenta. Os componentes gráficos como Blocos, Conexões e Diagramas baseiam-se na biblioteca Goocanvas[3].

Camada de Modelo: representam os componentes do sistema como Diagrama, Porta, Conexão, Bloco, Comentário e Padrão de Código. Um Bloco pode possuir portas de Entrada e de Saída e uma Conexão é uma relação dois Blocos e suas respectivas Portas, conforme ilustrado na Figura 3.

4.1. Estendendo a arquitetura por *plugins*

O Mosaicode pode ser estendido por meio da criação de novos Blocos, Portas e Padrões de Código e traz um gerenciador para a criação e gerenciamento destas extensões. Com isto, usuários mais experientes podem criar novos trechos de código ou alterá-los conforme suas necessidades. Uma **extensão** pode ser feita tanto como uma classe Python quanto como um arquivo XML. O carregamento das extensões se dá na seguinte ordem: Classes Python instaladas com o sistema, Arquivos XML instalados com o sistema e Arquivos XML no espaço de usuário. Assim, se o usuário quiser alterar e personalizar Blocos em sua instância da ferramenta, o mesmo consegue fazer de maneira a alterar a instalação da ferramenta apenas para seu usuário e sem depender de senha de acesso especial para isto.

Além disto, a ferramenta suporta também a adição de *plugins* para inserir novas funcionalidades no sistema. Desta forma, a ferramenta pode ser configurada para ter funcionalidades distintas dependendo do perfil do usuário final. O Gerenciador e Criador de extensões é um exemplo de *plugin* da ferramenta e que pode não ser instalado com o Mosaicode por usuários iniciantes.

5. Ferramentas relacionadas

Entre as ferramentas relacionadas com este projeto estão os ambientes de programação visual Pure Data[4], um ambiente de programação visual para som e música que também serve de hospedeiros para o ambiente GEM de processamento de gráfico 3D [Puckette et al. 1996, Danks 1997]; o Max/MSP[5] que também é um ambiente de programação visual para processamento de imagem, som e vídeo [Wright et al. 1999]; e o Eyesweb[6], um ambiente de programação visual para imagem, som e vídeo, mas focado na análise de gestos e no movimento do corpo [Camurri et al. 2000]. Estes ambientes de programação visual simplificam a criação de ferramentas para artistas, porém não possibilitam o acesso ao código fonte pelo usuário.

[3]Website do projeto: `https://wiki.gnome.org/Projects/GooCanvas`.
[4]Website do projeto: `http://puredata.info`.
[5]Website do proketo:`https://cycling74.com/products/max`.
[6]Website do projeto: `http://www.infomus.org/eyesweb_ita.php`.

O Processing[7] é uma linguagem textual para o domínio de Arte Digital desenvolvido no Media Lab do MIT [Reas and Fry 2007] focada primordialmente em Arte gráfica que inclui uma DSL, uma IDE e um gerador de código. Outra linguagem DSL que trabalha com geração de código no contexto da computação musical chama-se Faust [Orlarey et al. 2009], uma linguagem de programação funcional para processamento de sinais musicais que gera código para diferentes linguagens, bibliotecas e *API's*.

A ideia de unir VPL com geração de código é também explorada pelo EarSketch [Mahadevan et al. 2016], um ambiente de programação visual baseado em web e usado para criar aplicações baseado no ambiente de programação Blockly [Fraser 2012].

6. Exemplos de uso

Processamento de imagens e visão computacional: Seguindo a ideia do projeto Harpia, o Mosaicode conta com uma extensão para processamento de imagens e visão computacional baseado na biblioteca OpenCV na linguagem C++. Esta extensão conta com blocos para operações lógicas, morfológicas e aritméticas de imagens, tipos básicos de dados, formas básicas, experimentais, filtros e conversão de cores além de detecção de objetos, pessoas e faces.

Web Art: O Mosaicode conta com uma extensão para o desenvolvimento de aplicações musicais em javascript cujos objetos são baseados na API webaudio do HTML5. Isto inclui osciladores, ruído, microfone e também efeitos de áudio que permitem transformar o navegador Web em um ambiente de programação musical [Roberts et al. 2013].

Síntese de imagens: Para síntese de imagens foi desenvolvido uma extensão utilizando a biblioteca OpenGL [Woo et al. 1999] para C. Inicialmente, foram criados blocos representando formas geométricas 2D básicas, como Círculo, Quadrilátero e Triângulo, e formas 3D nativas do OpenGL, como Cubo, Esfera, Cone e Teapot. Após esta etapa inicial, foram implementadas operações para transformação do objeto, como Translate, Rotate e Scale, e operações de Push e Pop, para definir quais das formas sofrerão alteração.

Síntese de som: Para síntese de som na linguagem C foi desenvolvido uma extensão que utiliza a *API* PortAudio para o acesso ao dispositivo de som do computador. Esta extensão possui um conjunto de Blocos para criação de aplicações musicais e composição.

7. Potenciais usuários

Entre as áreas que fazem historicamente uso de Programação Visual está a arte digital [Hils 1992], na qual possibilita a criação de artes multimídia em um espaço virtual. A criação artística digital se assemelha com a criação de ambientes de realidade virtual [Schiavoni and Gonçalves 2017] e também a área de jogos, especialmente no que tange o aparato técnico necessário para esta criação.

8. Conclusão

O ambiente de programação visual Mosaicode aqui apresentado vem se mostrando eficiente para a criação artística digital e para o ensino das técnicas envolvidas nesta criação como a Computação Musical, Processamento Digital de Imagens, Computação Gráfica, Visão Computacional e Realidade Virtual [Schiavoni and Gonçalves 2017]. Diferente dos outros ambientes de programação visual aqui apresentados, o Mosaicode é um gerador de código e atualmente possui extensões para a geração de código nas linguagens Javascript, C e C++ usando as *API's* webaudio, openGL e openCV. Assim, mais do que ensinar a programar visualmente, este ambiente possibilita a prototipação rápida de aplicações sendo que o acesso ao código gerado permite a otimização posterior do código desenvolvido.

[7]Website do projeto: `https://processing.org/`.

Uma característica importante deste ambiente é a capacidade de instalação de novos *plugins* e extensões construídos no próprio ambiente, proporcionando uma maior flexibilidade à ferramenta de modo que os usuários possam customizá-la de acordo com as suas necessidades.

Referências

Camurri, A., Hashimoto, S., Ricchetti, M., Ricci, A., Suzuki, K., Trocca, R., and Volpe, G. (2000). Eyesweb: Toward gesture and affect recognition in interactive dance and music systems. *Computer Music Journal*, 24(1):57–69.

Danks, M. (1997). Real-time image and video processing in gem. In *ICMC*.

Fraser, N. (2012). Google blockly-a visual programming editor. *URL: http://code. google. com/p/blockly, accessed Aug*.

Grau, O. (2003). *Virtual Art: from illusion to immersion*. MIT press.

Hils, D. D. (1992). Visual languages and computing survey: Data flow visual programming languages. *Journal of Visual Languages & Computing*, 3(1):69–101.

Mahadevan, A., Freeman, J., and Magerko, B. (2016). An interactive, graphical coding environment for EarSketch online using Blockly and Web Audio API. In *Web Audio Conference WAC*. Georgia Institute of Technology.

Orlarey, Y., Fober, D., and Letz, S. (2009). FAUST: an efficient functional approach to DSP programming. *New Computational Paradigms for Computer Music*, 290.

Puckette, M. (2002). Max at seventeen. *Computer Music Journal*, 26(4):31–43.

Puckette, M. et al. (1996). Pure data: another integrated computer music environment. *Proceedings of the second intercollege computer music concerts*, pages 37–41.

Reas, C. and Fry, B. (2007). *Processing: a programming handbook for visual designers and artists*. Number 6812. Mit Press.

Roberts, C., Wakefield, G., and Wright, M. (2013). The Web Browser As Synthesizer And Interface. In *NIME*, pages 313–318. Citeseer.

Schiavoni, F. L. and Gonçalves, L. L. (2017). From Virtual Reality to Digital Arts with Mosaicode. In *2017 19th Symposium on Virtual and Augmented Reality (SVR)*, pages 200–206, Curitiba - PR - Brazil.

Schiavoni, F. L. and Gonçalves, L. L. (2017). Teste de Usabilidade do Sistema Mosaicode. In *Anais [do] IV Workshop de Iniciação Científica em Sistemas de Informação (WICSI)*, pages 5–8, Lavras - MG: Universidade Federal de Lavras - UFLA.

Wands, B. (2007). *Art of the Digital Age*. Thames & Hudson.

Woo, M., Neider, J., Davis, T., Shreiner, D., et al. (1999). *OpenGL programming guide*, volume 3. Addison-wesley Reading.

Wright, M., Dudas, R., Khoury, S., Wang, R., and Zicarelli, D. (1999). Supporting the Sound Description Interchange Format in the Max/MSP Environment. In *ICMC*.