playAPC: Uma Biblioteca Gráfica para Programadores Iniciantes

Sinayra P. C. Moreira[1]* Alexandre Zaghetto[1] José C. L. Ralha[1] Mateus Mendelson[2]

[1]Universidade de Brasília, Departamento de Ciência da Computação, Brasil
[2]Universidade de Brasília/FGA, Departamento de Engenharia de Software, Brasil

Figura 1: Biblioteca playAPC.

Resumo

Este artigo apresenta a biblioteca gráfica playAPC, desenvolvida para que programadores iniciantes possam utilizar modelagem gráfica como ferramenta para consolidar conceitos e fundamentos adquiridos já no primeiro semestre de curso, especificamente na disciplina *Algoritmos e Programação de Computadores* da Universidade de Brasília. O fator estimulante baseia-se no desenvolvimento de jogos, dos mais simples aos mais complexos. Espera-se com isso aumentar o interesse pelas disciplinas avançadas de computação, bem como as matérias relacionadas das áreas de matemática e física.

Keywords: Algoritmos e Programação de Computadores, Programação para Iniciantes, Computação Gráfica, Jogos.

1 Introdução

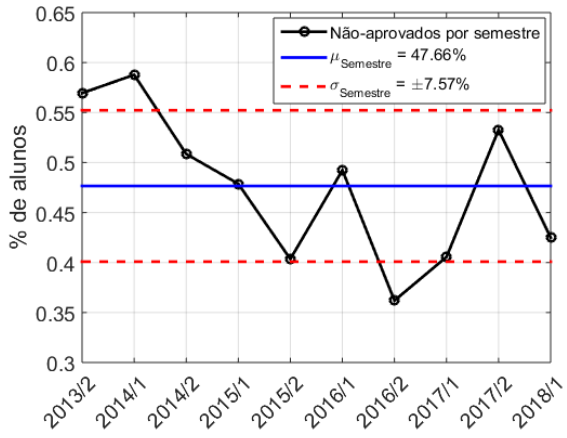
Parte-se da premissa de que a educação científica e tecnológica deve ser considerada como elemento estratégico dentro do plano de desenvolvimento de um país e que deve ser iniciada o mais cedo possível [1]. Os efeitos de tal educação são: (a) mão-de-obra capacitada a ocupar postos de trabalho que requerem conhecimentos tecnológicos avançados; (b) inovações que contribuem para o bem-estar social e geram renda para o país; e (c) uma população capaz de pensar criticamente a respeito dos fenômenos naturais que a cercam. Entre as áreas de conhecimento que podem ser consideradas em um projeto de educação científica e tecnológica estão: Física, Química, Matemática, Biologia, Ciência da Computação, Engenharia de Computação, Engenharia Elétrica, Engenharia Eletrônica, Engenharia Mecânica, Engenharia Civil, Engenharia Acústica, Engenharia Aeroespacial, Astronomia, Nanotecnologia, Física Nuclear, Robótica, Bioquímica, Biomecânica e Bioinformática. Essa lista não esgota todas as possibilidades, mas representa um conjunto de disciplinas bastante significativo no contexto deste documento.

*e-mail: sinayra@outlook.com

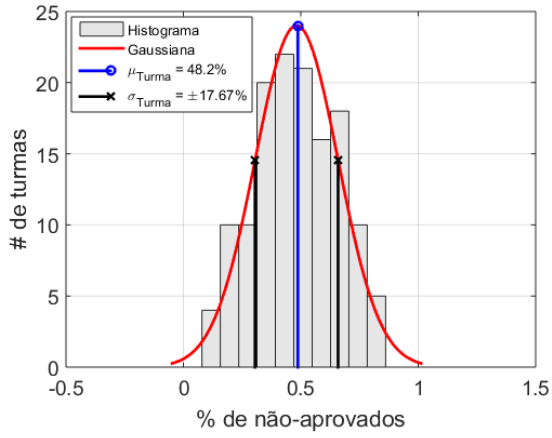
Em especial, o papel que a Ciência da Computação exerce atualmente sobre a vida dos cidadãos é inquestionável e apresenta um caráter progressivo [2]. Apenas em um cenário muito improvável a Computação perderia importância enquanto área de conhecimento. Seu impacto na saúde, no transporte, na segurança, na educação, na cultura, no entretenimento e nas telecomunicações definiram uma nova forma de organização do ser humano e tudo indica que seu avanço será cada vez mais necessário tendo-se em vista o objetivo maior de se construir uma sociedade mais justa.

Num cenário em que a Computação ganha cada vez mais importância, saber solucionar problemas a partir de um perspectiva algorítmica é uma habilidade que não deve estar restrita aos que se dedicam à Computação como atividade fim. Vários países, tendo percebido isso, optaram por incorporar a Programação de Computadores ao currículo obrigatório do ensino básico [3, 4], dando a essa disciplina um *status* semelhante ao da Matemática ou do Português, por exemplo. Assume-se, ainda, que os alunos que seguirem uma carreira científica ou tecnológica já terão tido contato anterior com os princípios da programação, o que irá potencializar seu aprendizado no ensino superior. No entanto, enquanto isso não é uma realidade, cabe aos cursos superiores introduzirem o assunto a seus alunos.

Nos cursos de computação e correlatos, a primeira matéria de algoritmos é fundamental para que o aluno possa desenvolver suas habilidades como programador. Como forma de potencializar o desenvolvimento do pensamento computacional, o Departamento de Ciência da Computação da Universidade de Brasília elaborou uma biblioteca baseada em computação gráfica, denominada playAPC [5], que possui a identidade visual apresentado na Figura 1. A playAPC foi desenvolvida para ser utilizada em matérias de iniciação à programação, como, por exemplo, a matéria Algoritmos e Programação de Computadores (APC), que é ofertada para os cursos de Engenharia e Computação. A biblioteca trata-se de uma ferramenta educacional voltada para programadores pouca ou nenhuma experiência e possibilita a visualização de conceitos teóricos por meio de elementos de computação gráfica. Com ela, é possível criar desenhos estáticos, animações simples e até mesmo jogos. Ela consiste em um conjunto de funções para criação e manipulação de formas 2D e utilização de imagens. Como recurso



(a)



(b)

Figura 2: Índices de não-aprovação em APC ao longos dos últimos 5 anos. (a) O gráfico mostra a porcentagem de não aprovação por semestre, em preto, a porcentagem média de reprovação nos semestres analisados, μ em azul, e intervalo compreendido entre mais e menos um desvio padrão da média, $\mu \pm \sigma$ em vermelho. (b) O histograma apresenta a taxa de reprovação por turma, tendo uma aproximação Gaussiana, em vermelho, uma média de reprovação μ e um desvio padrão $\mu \pm \sigma$.

gráfico, é utilizada a API OpenGL[6]. Todo o seu desenvolvimento foi realizado com a linguagem de programação C++[7]. Caso o curso de APC seja ministrado utilizando-se a linguagem C [8, 9], não haverá quaisquer prejuízos no aprendizado da disciplina, pois a biblioteca aqui apresentada possui interfaces totalmente compatíveis com a linguagem C.

Apesar de sua grande importância, o aprendizado da programação não é trivial. A Figura 2 mostra os índices de não-aprovação na principal disciplina de iniciação à programação oferecido aos cursos de Engenharia e Computação da Universidade de Brasília, por semestre, ao longo dos últimos cinco anos. Durante esse intervalo de tempo, foi atendido um total de 6057 alunos e observou-se uma taxa de não-aprovação de 36% à 58%, variando ao longo dos semestres. A média de não-aprovação por turma foi de aproximadamente 48.2% com um desvio padrão de 17.7%. Tais índices são preocupantes, pois esse é o momento em que o aluno deve estabelecer as bases de suas habilidades como programador e consequentemente obter sucesso nas demais etapas do curso. Apesar das tentativas de se criar mais horários de plantão de dúvidas e maior disponibilidade dos monitores para a disciplina, as taxas de não-aprovação permanecem em patamares preocupantes. É importante ressaltar que aprovado foi considerado o aluno que obteve menção CC (aproveitamento de créditos), MM (nota 5 à 6,99), MS (nota 7 à 8,99) e SS (nota 9 à 10); e não-aprovado, o aluno cuja menção foi MI (nota 3 à 4,99), II (nota 1 à 2,99), SR (nota 0 ou excedeu limite de faltas), TR (trancamento sem justificativa) e TJ (trancamento com justificativa).

Sendo assim, o presente trabalho apresenta a biblioteca gráfica playAPC, cujo objetivo é dar suporte às políticas pedagógicas da área de fundamentos de programação, aumentando o engajamento do aluno pela disciplina inicial de programação. Mais especificamente, pretende-se mostrar que a biblioteca proposta pode ser utilizada no desenvolvimento de jogos, um importante segmento de mercado que costuma atrair a atenção dos alunos ingressantes. Se a sequência de aulas for adequadamente planejada com o uso da playAPC, ao final do semestre letivo os alunos deverão ser capazes de programar jogos simples.

2 A Biblioteca playAPC

Do ponto de vista do aluno, a playAPC serve como ferramenta de consolidação dos conceitos abordados em disciplinas de iniciação à programação. A modelagem gráfica por ela fornecida tem como objetivo tornar mais prazeroso o aprendizado da programação. Para o docente, a biblioteca, além de permitir ilustrar visualmente os conteúdos da disciplina, oferece maior liberdade criativa na elaboração de práticas de laboratórios, diversificando os tipos de problemas a serem resolvidos. Com o auxílio da ferramenta proposta, os problemas podem envolver a manipulação de imagens e jogos simples, objeto de discussão do presente artigo.

Pesquisando ementas de outras universidades que oferecem o curso de computação, foram encontradas cinco que, em sua primeira matéria de programação, permitem ao aluno a visualização gráfica de seus programas, seja durante todo o semestre ou só em parte dele [1]. Além disso, levou-se também em consideração qual linguagem é utilizada nestes cursos. A decisão a respeito de qual linguagem de programação utilizar no primeiro contato com programação é bastante controversa, mas com bastante frequência a linguagem C é empregada. Recentemente, uma publicação da IEEE Spectrum[10] apresentou um ranking mostrando a popularidade dessa linguagem, que figura na segunda posição, atrás apenas do Python e antes do Java. Muitos dizem que para um primeiro curso de algoritmos e programação de computadores, o C é demasiadamente difícil. O argumento da dificuldade, porém, não parece válido, pois, em primeiro lugar, é possível reduzir o grau de dificuldade do uso dessa linguagem quase ao ponto em que ela começa a se tornar semelhante a linguagens "*mais fáceis*". Na verdade, se o foco principal é de fato o desenvolvimento de algoritmos, as diferenças entre o uso simplificado do C e o uso de qualquer outra linguagem mais simples são reduzidas a quase nenhuma. Em segundo lugar, graus de dificuldade mais elevados, se abordados com a didática adequada, podem aperfeiçoar a capacidade cognitiva e a criatividade do aluno. Ou seja, um uso mais elaborado da linguagem C pode propiciar ganhos cognitivos úteis também a elaborações algorítmicas. Um recurso de programação mais complexo pode permitir a proposição de soluções algorítmicas também mais complexas. Seja como for, se a linguagem C não for empregada, uma outra que apresente fortes semelhança com o C seria indicado, pois, assim, serviria como introdução não apenas à referida linguagem, mas a toda uma família de linguagens que herdaram grande parte de suas características.

Considerando que playAPC foi desenvolvida para ser utilizada em conjunto com a linguagem C/C++, buscaram-se alternativas

[1] `http://goo.gl/N5sDgY`

para o uso de modelagem gráfica nos cursos de iniciação à programação que utilizassem a mesma linguagem e encontraram-se cinco alternativas: (a) PlayLib, desenvolvida na PUC/Rio[11]; (b) WinBGIm[12]; (c) API SDL[13]; (d) Turtle Grafik, desenvolvida na ETH Zürich[14] e; (e) OpenFrameworks [15].

Apesar das bibliotecas e *APIs* listadas possuírem o mesmo propósito, tornar a aprendizagem de programação mais prazeroza utilizando recursos gráficos na linguagem C, cada uma delas não abrange todas as funcionalidades que a playAPC possui. Comparando cada uma com a playAPC, podemos destacar as seguintes diferenças: (a) A PlayLib está restrita ao sistema operacional Windows e exige do aluno o conceito de *callback* logo no seu primeiro contato com a biblioteca. A playAPC possui guia de instalação para Windows, Linux da distribuição Debian, Mac OSX e não exige nenhuma abstração em seu primeiro uso além de algoritmos sequenciais, com as funções *AbreJanela*, *Desenha* e as funções para criação de geometrias; (b) A WinBGIm está restrito ao sistema operacional Windows e não possui suporte nativo para animação, forçando o aluno a fazer animação quadro à quadro, ou seja, renderizar uma cena, apagá-la, renderizá-la de novo com leves diferenças em relação a cena anterior e assim sucessivamente. A playAPC encapsula o laço de renderização com as funções *Desenha* e *Desenha1Frame*; (c) A SDL não encapsula o laço de renderização, exigindo que o aluno saiba no seu primeiro contato com a *API* o conceito de ponteiros; (d) A Turtle Grafik renderiza apenas figuras geométricas, não tendo suporte à leitura de imagens. A playAPC realiza leitura de imagem em qualquer extensão com a função *AbreImagem* e a renderiza em uma geometria com a função *AssociaImagem*. (e) A OpenFrameworks possui uma gama muito maior de funcionalidades que a própria playAPC, com funcionalidades mais complexas e exigindo do aluno um entendimento prévio de computação gráfica para conseguir usar toda a potencialidade do framework, tornando-o uma proposta menos enxuta para ser aplicada na matéria inicial de computação. Além disso, a OpenFrameworks exige, em seu primeiro contato com o framework, o entendimento de orientação à objetos. A playAPC possui a proposta de ser utilizada em sua capacidade total no término do primeiro semestre.

Suprindo todos os pontos levantados, a playAPC ainda possui licença GNU GPL, permitindo a manutenção do código não apenas pelos autores originais, mas também pela comunidade acadêmica. Isto possibilita a futura incorporação de mais funcionalidades além das apresentadas neste artigo, como por exemplo, a renderização de texto ou a manipulação de áudio.

Com a playAPC, é possível elaborar atividades que cubram os principais tópicos de APC: algoritmos sequenciais, algoritmos condicionais e com repetição, além de vetores, matrizes e estruturas. Para dar suporte ao uso da biblioteca foram elaborados dois materiais de apoio: um guia de referência e uma apostila. O Guia de Referência [2] disponibiliza ao usuário breves explicações sobre cada função da playAPC, exemplificando o seu uso. No guia, também está disponível detalhadamente instruções de instalação da biblioteca para os sistemas operacionais Windows, Linux e Mac.

A apostila [3] disponibiliza uma bateria de problemas que podem ser resolvidos graficamente, separados por tópicos. Com esse material, é possível ao docente organizar suas aulas práticas de laboratório com exemplos de uso da playAPC, contando também com exercícios desafios, os quais envolvem todos os tópicos tratados no documento.

Nessa Seção, as potencialidades da biblioteca serão apresentadas por meio de quatro exemplos. São eles: (a) Carro em Movimento; (b) Filtragem de Média; (c) Gerador Sub-kandinsky; e (d) Gêiser. Cada exemplo será detalhado a seguir.

[2] http://playapc.zaghetto.com/

[3] https://github.com/sinayra/playAPC-livro/

2.1 Carro em Movimento

Esta aplicação tem por objetivo mostrar na tela do computador um carro estilizado, desenhado pelo próprio programador a partir das funções básicas de desenho da biblioteca, que se desloca de um canto ao outro da janela de contexto (espaço visual onde a aplicação gráfica é exibida). Inicialmente, o carro deve ser posicionado no canto central esquerdo da janela de contexto e se mover em direção ao canto central direito da janela, deslocando-se passo a passo. Para que o experimento seja realizado, é necessário o uso de laços de repetição (*for* ou *while* ou *do...while*), que são o foco da atividade.

Primeiramente, é necessário incluir o *header* da playAPC no código do programa. A Listagem 1 mostra como realizar a inclusão, assumindo-se que a biblioteca tenha sido corretamente instalada seguindo o Guia de Referência.

```
#include <playAPC/playapc.h>
```

Listagem 1: Inclusão da playAPC no código.

Uma variável do tipo *Ponto* é necessária para indicar o local, no plano cartesiano, em que cada geometria será criada. A Listagem 2 destaca os tipos de variáveis a serem utilizadas.

```
Ponto p;
int carro;
```

Listagem 2: Declaração de variáveis.

Após a declaração das variáveis, a primeira função a ser chamada é aquela que abre a janela de contexto, a *AbreJanela*. Essa função recebe dois valores inteiros que representam a largura e a altura da janela em *pixels*, além de uma *string* que indica o título da janela. Por padrão, o fundo da janela é da cor preta, o que pode ser alterado por meio da função *PintarFundo*. Essa função recebe como parâmetro de entrada três valores inteiros que representam a cor no espaço de cores *RGB*. A Listagem 3 cria uma janela de 400 *pixels* de altura, 400 *pixels* de altura, com o nome *Carrinho* e com o fundo branco.

```
AbreJanela(400, 400, "Carrinho");
PintarFundo(255, 255, 255);
```

Listagem 3: Abertura da janela de contexto.

Uma vez que a atividade consiste em uma animação, que realiza um movimento de translação, é preciso utilizar uma variável para armazenar o conjunto de geometrias que compõem o carro. A variável *carro* é utilizada para esse fim. Ela deve ser inicializada com o retorno da função *CriaGrupo*. Essa função retorna um índice de um conjunto de geometrias. Inicialmente, esse conjunto começa vazio e todas as geometrias que forem criadas após a chamada dele irão pertencer à ele. A ordem de declaração de geometrias faz diferença por se tratar de geometrias em um plano de duas dimensões. Internamente, a playAPC utiliza esse índice como início da lista de geometrias, sempre inserindo no final da lista a cada nova geometria criada. Porém, para o aluno, basta interpretar este grupo como um índice que se refere ao conjunto de geometrias e que cada geometria criada após a chamada *CriaGrupo* pertencerá a este conjunto. A Listagem 4 mostra como inicializar esse novo conjunto.

```
carro = CriaGrupo();
```

Listagem 4: Criação de um conjunto.

Após a inicialização da variável *carro*, é possível declarar todas as geometrias que pertencem ao conjunto. No presente trabalho, será utilizado um retângulo para a criação do corpo e do capô do carro. Um retângulo na playAPC é definido com um valor inteiro que representa a *base*, um valor inteiro que representa a *altura* e um *ponto de referência p* do tipo *Ponto*. O *ponto de referência* indica o ponto a partir do qual o canto inferior esquerdo do retângulo será criado. Sabendo que o ponto $(0,0)$ representa o centro da janela, o

corpo do carro pode ser criado, por exemplo, no ponto $(-100, 20)$. A variável *p* é uma estrutura que possui duas coordenadas, *x* e *y*, que correspondem à uma coordenada do plano cartesiano. Somente após a criação da geometria a função *Pintar*, que atribui uma cor à geometria, pode ser chamada. Essa função recebe como entrada três valores inteiros, indicando a cor no padrão *RGB*. A Listagem 5 exemplifica o uso da função *CriaRetangulo*.

```
/*corpo do carro*/
p.x = -100;
p.y = 20;
CriaRetangulo(30, 20, p);
Pintar(128, 0, 0);

/*capô do carro*/
p.x = -80;
p.y = 20;
CriaRetangulo(20, 12, p);
Pintar(128, 0, 0);
```

Listagem 5: Determinação da base do carro.

Para criar as rodas do carro, serão utilizados dois círculos pintados de preto. Um círculo na playAPC é definido com um valor inteiro que indica o *raio* e um *ponto de referência p* do tipo *Ponto*. O *ponto de referência* indica o centro do círculo. A Listagem 6 exemplifica o uso da função *CriaCirculo*.

```
/*roda 1*/
p.x = -95;
p.y = 20;
CriaCirculo(4, p);
Pintar(0, 0, 0);

/*roda 2*/
p.x = -75;
p.y = 20;
CriaCirculo(4, p);
Pintar(0, 0, 0);
```

Listagem 6: Determinação das rodas do carro.

Com todas as geometrias criadas, o próximo passo é criar o processo de animação, o que é feito por meio da função *Move*. A função *Move* recebe como parâmetros um *grupo* de geometrias e um *ponto de referência*. Essa função redesenha o grupo em uma nova posição da janela de contexto. O *ponto de referência* se refere ao ponto da primeira geometria criada no grupo. No exemplo aqui considerado, refere-se ao corpo do carro. Como nenhuma alteração foi feita em relação aos limites do plano cartesiano, ele está limitado em 200 unidades, cujos valores vão de -100 até 100, tanto no eixo *x* quanto no eixo *y*. Independentemente do redimensionamento e do tamanho em *pixels* da janela, a existência de 200 unidades será sempre mantida de forma proporcional.

Após movimentar o grupo para a próxima posição, a nova cena é deve ser renderizada por meio da função *Desenha1frame*, que renderiza somente uma cena com todas as geometrias e transformações realizadas. A Listagem 7 ilustra como o bloco de repetição *for* é utilizado para movimentar o carro de uma em uma unidade.

```
p.y = 20;
for(p.x = -100; p.x < 100; p.x++){
  Move(p, carro);
  Desenha1Frame();
}
```

Listagem 7: Bloco de repetição para a animação.

Assim que o carro chega ao seu destino final, podemos renderizar o resultado final com a função *Desenha*, que renderiza a cena de forma estática à uma taxa de 30 quadros por segundo, em um laço infinito. Não é possível encerrar essa função a menos que o usuário encerre o programa fechando a janela de contexto da playAPC.

```
Desenha();
```

Listagem 8: Renderização da última cena.

A Figura 3 exibe o resultado final e o código fonte pode ser consultado na íntegra na Listagem 9.

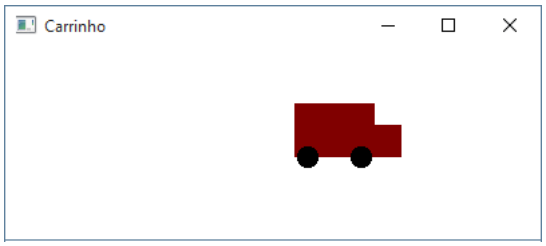


Figura 3: Carro percorrendo a tela.

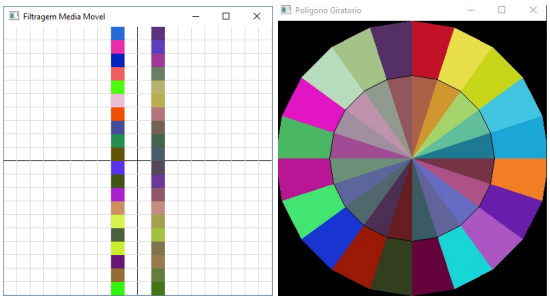
```
#include <playAPC/playapc.h>

int main(){
  Ponto p;
  int carro;

  AbreJanela(400, 400, "Carrinho");
  PintarFundo(255, 255, 255);

  carro = CriaGrupo();
  /*corpo do carro*/
  p.x = -100;
  p.y = 20;
  CriaRetangulo(30, 20, p);
  Pintar(128, 0, 0);

  /*capô do carro*/
  p.x = -80;
  p.y = 20;
  CriaRetangulo(20, 12, p);
  Pintar(128, 0, 0);

  /*roda 1*/
  p.x = -95;
  p.y = 20;
  CriaCirculo(4, p);
  Pintar(0, 0, 0);

  /*roda 2*/
  p.x = -75;
  p.y = 20;
  CriaCirculo(4, p);
  Pintar(0, 0, 0);

  p.y = 20;
  for(p.x = -100; p.x < 100; p.x++){
    Move(p, carro);
    Desenha1Frame();
  }
  Desenha();
}
```

Listagem 9: Exemplificação do uso de *for* com um carro.

2.2 Filtragem de Média

Além do exercício básico apresentado na Seção 2.1, que ilustra o uso de laços de repetição, o exercício de Filtragem de Média ilustra o conceito de vetores, sendo aplicado em geometrias coloridas que utilizam o espaço de cores *RGB*. A prática consiste em gerar

aleatoriamente 20 componentes *RGB*, exibi-los na tela, aplicar a filtragem de média e mostrá-los lado a lado para comparação com as cores originais. A Figura 4 apresenta um exemplo de resultados: (a) o que se espera dos alunos, ou seja, duas colunas de componentes, uma com as cores originais e outra com o valor filtrado; e (b) uma proposta de apresentação alternativa, sendo composta por um círculo exterior com as cores originais e um círculo interior com as cores filtradas, exibindo os componentes de maneira visualmente mais interessante e criativa. Essa última apresentação foi proposta pelo aluno Pedro Augusto Coelho Nunes. Além de mostrar o resultado da filtragem, ele implementou uma animação na qual o círculo interno gira no sentido horário e a coroa externa gira no sentido anti-horário.



(a) Resultado esperado. (b) Resultado proposto.

Figura 4: Exercício de Filtragem de Média

2.3 Gerador Sub-Kandinsky

Apesar de sua proposta é ser utilizada em aulas de laboratório da matéria de iniciação à programação, a playAPC pode ser aplicada em diversos contexto onde a exibição em um plano 2D é requerida. No contexto artístico, a Figura 5 ilustra um exemplo de imagem criada pseudo-aleatoriamente pelo Gerador Sub-Kandinsky, que tenta imitar a obra *Circles in Cicle*, do pintor Wassily Kandisky. A cada execução deste gerador, ele produz formas e cores seguindo regras induzidas após análise da obra original.

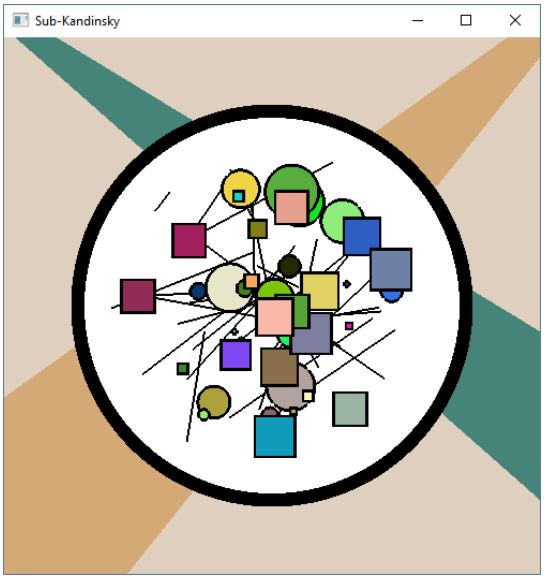


Figura 5: Gerador Sub-Kandinsky.

2.4 Gerador Gêiser

A Figura 6 representa um exemplo de relevo gerado pseudo-aleatoriamente, onde há um gêiser que teve sua posição escolhida também de forma pseudo-aleatória. O relevo é definido por tons entre o verde e o amarelo, sendo que o verde mais puro representa planícies e o amarelo mais puro representa planaltos. A cada iteração, o gêiser vai inundando a região, despejando mais água a partir da sua posição inicial e acrescentando mais *blocos de água* nas regiões que já estão inundadas, respeitando a altura de cada terreno. A cada vazão de novos *blocos de água*, o tom da água torna-se puro. A execução se encerra quando todo o mapa é inundado.

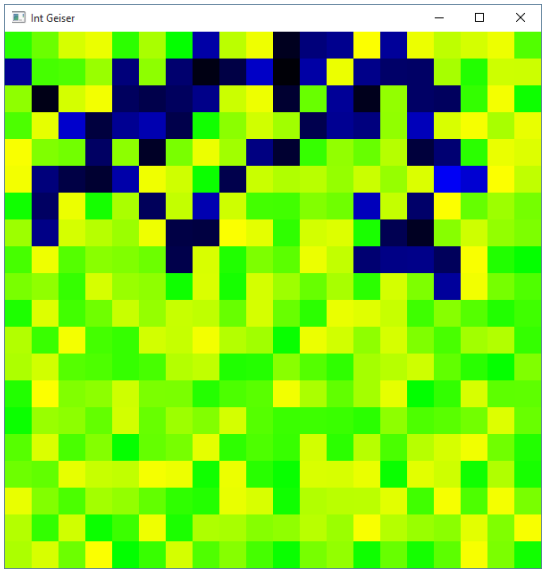


Figura 6: Gerador Gêiser.

3 Aplicações Didáticas da playAPC no Desenvolvimento de Jogos

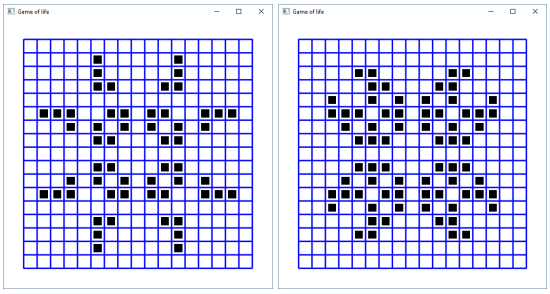
Nessa Seção, as potencialidades da biblioteca em aplicações voltadas ao desenvolvimento de jogos serão exemplificadas nas práticas: (a) jogo da vida; (b) Corrida de Obstáculos; (c) Snake; (d) Batalha Naval de Um Jogador; e (e) APCway.

Tendo o desenvolvimento de jogos como fator estimulante para incentivar o estudo de programação, podemos começar com jogos que não lidam com nenhum tipo de evento, como o Jogo da Vida, até chegar em jogos com diversos tipos de eventos, como APCway. Esta Seção exibe alguns dos exemplos de jogos feitos com a playAPC em ordem crescente de complexidade.

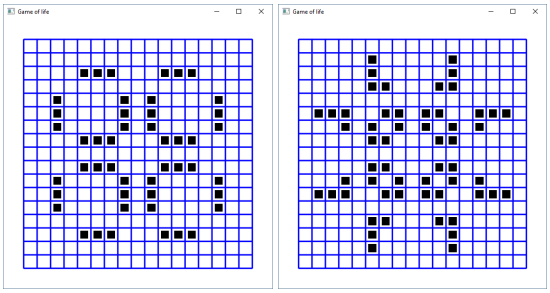
3.1 Jogo da Vida

O jogo da vida é um autômato que opera em uma matriz de duas dimensões com células quadradas, proposto pelo matemático John Conway por volta de 1970 [16]. A partir de uma dada população inicial, distribuída nesta matriz, as próximas gerações são determinadas de acordo com quatro regras:

1. Qualquer célula com menos de dois vizinhos morre por sub-população;
2. Qualquer célula com dois ou três vizinhos continua viva;
3. Qualquer célula com mais de três vizinhos morre por super-população; e
4. Qualquer célula morta com exatamente três vizinhos torna-se viva por reprodução.



(a) Primeira geração. (b) Segunda geração.

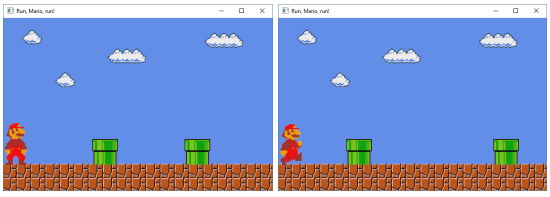


(c) Terceira geração. (d) Quarta geração.

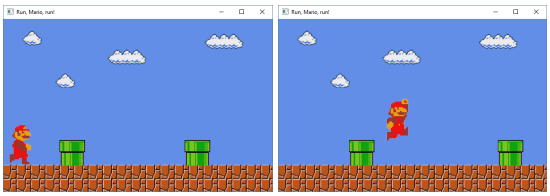
Figura 7: Exemplo de um padrão de jogo da vida: Pulsar.

O Jogo da Vida, como foi proposto inicialmente, possui características de um simulador, simulando gerações de células que nascem e morrem, não tendo condições de vitória ou derrota nem tampouco interação com o usuário. Trata-se de um jogo com zero jogadores. A Figura 7 exibe as gerações do Pulsar, onde observa-se um a repetição dp padrão de população após 3 iterações.

3.2 Corrida de Obstáculos



(a) (b)



(c) (d)

Figura 8: Corrida de obstáculos: (a) Personagem parado; (b) *Sprite* 1 de andando; (c) *Sprite* 2 de andando; (d) Personagem pulando.

O programa Corrida de Obstáculos é similar ao Carro em Movimento, apresentado na Seção 2.1. A diferença entre esses dois jogos consiste no fato de que, em Corrida de Obstáculos, as posições dos canos são sorteadas aleatoriamente. Após o sorteio, o jogo é iniciado, consistindo em um personagem principal que detecta automaticamente os canos e pula por cima deles.

A Corrida de Obstáculos também possui características de um simulador, onde o personagem somente percorre a tela, não tendo condição de derrota ou vitória, nem tampouco interação com o usuário. Porém, este exercício já manipula eventos de animação e utilização de imagens, tornando-o visualmente agradável. A Figura 8 mostra alguns estágios de animação deste simulador.

3.3 Snake

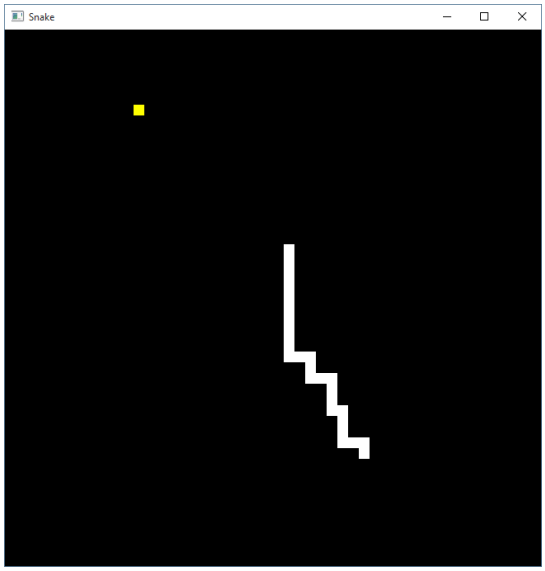


Figura 9: Jogo Snake.

Snake é um jogo onde se controla uma cobra que se move em velocidade constante com o objetivo de capturar o máximo possível de comida, posicionada aleatoriamente no mapa. A comida é representada por uma célula em amarelo, e só é capturada quando a cobra, representada por células brancas, colide com ela. Inicialmente a cobra começa com tamanho 1. A cada comida capturada, a cobra aumenta seu tamanho em uma célula, crescendo a partir da cauda. O jogo faz, então, um novo sorteio para determinar onde a próxima comida deve aparecer. O sorteio leva em consideração que células ocupadas pela cobra devem ser excluídas. O jogo é operado em uma matriz de duas dimensões com células quadradas e a Figura 9 mostra um estágio onde a cobra está com tamanho 18.

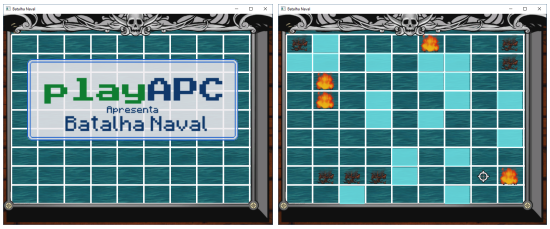
Apesar de não possuir telas de vitória ou derrota, Snake possui um claro objetivo e além de interação com o usuário, que controla a cobra via teclado.

3.4 Batalha Naval de Um Jogador

Batalha Naval de Um Jogador é um jogo onde se tenta adivinhar em que posições da matriz bidimensional o computador posicionou barcos, com o objetivo de afundá-los. A partir de um tabuleiro vazio de 9 linhas e 9 colunas, o computador deve posicionar aleatoriamente os barcos de acordo com as seguintes regras:

1. Existem 3 navios com comprimento igual a 3 posições do tabuleiro, 3 navios com comprimento igual a 2 posições e 4 navios com comprimento igual a 1;
2. Um navio não deve ocupar a posição de um outro, nem deve ser posicionado ao lado de outro: deve existir pelo menos um espaço vazio entre os dois navios;
3. Os navios podem estar nas verticais e nas horizontais, não nas diagonais;

4. Se um navio de comprimento igual a 1 for atingido, ele deve ser considerado como *afundado*;
5. Se um navio de comprimento igual a 2 ou 3 for atingido, ele deve ser considerado como *atingido*. Somente quando todas as peças deste navio forem atingidas é que deve-se indicá-lo como *afundado*; e
6. Se o usuário escolher uma posição do tabuleiro onde não há navio, deve-se indicar que o usuário atingiu água.



(a) Tela inicial. (b) Tela de batalha.

(c) Tela de vitória.

Figura 10: Telas de diferente estágios do jogo Batalha Naval de Um Jogador.

Este jogo possui um claro objetivo, interação com usuário, que controla o tiro com o mouse, e possui tela de vitória. A Figura 10 mostra diversos estágios do jogo.

3.5 APCway e o Certificado de Coragem

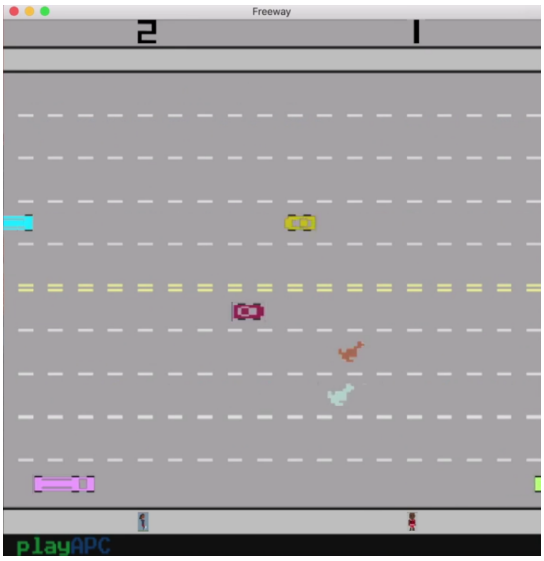


Figura 11: APCway para dois jogadores com diferentes obstáculos.

O jogo APCway é um jogo onde dois jogadores devem marcar o maior número de pontos fazendo uma galinha atravessar a rua. As cores dos obstáculos são geradas aleatoriamente e os obstáculos movem-se a uma velocidade constante com seu movimento inicializado em tempos aleatórios. Caso uma das galinhas seja atingida por um obstáculo, esta galinha volta para a calçada do canto inferior da tela. O jogador que juntar mais pontos ao final de um determinado tempo, ganha o jogo. A Figura 11 mostra uma implementação sugerida pelo aluno Pedro Caio Castro Cortes de Carvalho, onde os personagens são esportistas e há um obstáculo especial, a galinha gigante, que corre a uma velocidade diferente dos demais obstáculos. Apesar de não haver na biblioteca funções para manipulação de texto, o aluno implementou um placar de pontuação. Além disso, os personagens são controlados com botões implementados em uma placa Arduino. Assim como o jogo Batalha Naval de Um Jogador, APCway possui um claro objetivo, interação com usuário, tela de vitória e de derrota.

4 Conclusão

O presente trabalho apresentou uma biblioteca gráfica para ser aplicada em aulas de laboratório de disciplinas de iniciação à programação de computadores. Apesar de não ter sido realizada uma avaliação objetiva com professore e alunos, observou-se uma grande aceitação por parte de ambos os grupos. Os professores propuseram exercícios não listados na apostila, além de experimentarem a biblioteca em projetos pessoais que não estavam na ementa de suas disciplinas. Os alunos utilizaram os recursos da biblioteca para resolver os exercícios de forma criativa. Consideramos termos concluído com êxito o objetivo de desenvolver uma ferramenta que possa ser utilizada por programadores com pouca ou nenhuma experiência em computação gráfica.

Acompanhando o grau crescente da complexidade teórica da primeira matéria de programação, é possível estimular os alunos com práticas de rápido retorno lúdico, desenvolvendo jogos, dos mais simples aos mais complexos. No final do semestre, o aluno consegue desenvolver um jogo com alto grau de abstração, utilizando estruturas básicas de programação, como o APCway, apresentado na Seção 3.5. Outros exemplos de implementação deste jogo podem ser consultados no site da playAPC.

Para trabalhos futuros, pretende-se focar em instaladores para todos os sistemas operacionais suportados pela biblioteca. Apesar de não ter sido possível aplicar a playAPC em mais turmas de iniciação à programação, nas turmas ministradas pelos professores Alexandre Zaghetto e José Ralha, a principal queixa dos alunos foi o processo de instalação, especialmente para usuários de Windows e Mac. Para muitos, este é o primeiro contato de instalação de bibliotecas, APIs e até mesmo ambientes de desenvolvimento. Após este processo de instalação, a utilização se torna simples, já que é oferecido um guia de referência com as funções disponibilizadas pela playAPC. Além disso, deseja-se fazer uma pesquisa mais detalhada sobre o desempenho de turmas às quais a playAPCpode ser aplicada, incluindo turmas não apenas de cursos de computação, mas também de cursos correlatos, como Engenharia Civil, Matemática e outros. De acordo com os professores envolvidos, a adição de manipulação de objetos 3D também seria interessante, umas vez que a biblioteca foi desenvolvida em OpenGL, que visa manipular objetos 3D e 2D de forma eficiente.

Agradecimentos

Agradecemos ao professor José Carlos Loureiro Ralha que ofereceu o primeiro apoio e entusiasmo ao projeto, além do grande auxílio no desenvolvimento tanto na própria biblioteca como material didático. Também agradecemos a turma E e F do semestre 2016/2 de APC, que participou ativamente das práticas de laboratórios extras que utilizavam a playAPC. Em especial, agradecemos os alunos Pedro Augusto Coelho Nunes e Pedro Caio Castro Cortes de Carvalho, que forneceram os executáveis de suas aplicações para este trabalho.

Referências

[1] E. C. Förster, K. T. Förster, and T. Löwe. Teaching programming skills in primary school mathematics classes: An evaluation using game programming. In *2018 IEEE Global Engineering Education Conference (EDUCON)*, pages 1504–1513, Abril 2018.

[2] George Johnson. *The World: In Silica Fertilization; All Science Is Computer Science*. Acessado em 24 de julho de 2018.

[3] Charu Gusain. *After China and Japan, Canada is Planning To Teach Coding in Kindergarten*. Acessado em 07 de julho de 2018.

[4] Don Passey. *Computer science (CS) in the compulsory education curriculum: Implications for future research*, 2016. Eletronicamente `https://link.springer.com/article/10.1007/s10639-016-9475-z`.

[5] Sinayra Pascoal Cotts Moreira. PlayAPC : projeto e desenvolvimento de uma biblioteca gráfica como ferramenta didática para algoritmos e programação de computadores, 2016. Monografia (Bacharel em Ciência da Computação), UnB (Universidade de Brasília), Brasília, Brasil.

[6] Edward Angel and Dave Shreiner. *Interactive Computer Graphics*. Addison-Wesley, 2012.

[7] Kris Jamsa. *Programando em C/C++ A Bíblia*. Pearson, 1999.

[8] H. Deitel and P. Deitel. *C- Como Programar*. Pearson, 2011.

[9] B. W. Kernighan and D. M. Ritchie. *The C Programming Language*. Prentice Hall, 1988.

[10] Stephen Cass. The 2017 top programming languages. Eletronicamente, `http://sites.ieee.org/houston/article-2017-top-programming-languages/`. Acessado em 27 de julho de 2018.

[11] Edirlei Soares de Lima and Bruno Feijó. *Aprendendo a programar em C com playlib*, 2012. Acessado em 29 de setembro de 2016.

[12] Michael Main. Borland graphics interface (bgi) for windows. Eletronicamente, `http://winbgim.codecutter.org/`. Acessado em 02 de outubro de 2014.

[13] Lorraine Figueroa. *Beginner's guide to sdl*, 2011. Acessado em 30 de agosto de 2014.

[14] Felix Friedrich. Turtle grafik. Disponível em `http://lec.inf.ethz.ch/itet/informatik1/2015/`. Acessado em 18 de agosto de 2016.

[15] Theodore Watson Zach Lieberman and Arturo Castro. openframework. Eletronicamente, `https://openframeworks.cc`. Acessado em 12 de setembro de 2018.

[16] A. Adamatzky. *Game of Life Cellular Automata*, 2010. Springer News 7/8/2010.