IntEducaTrânsito: um jogo 3D interativo e educativo sobre as normas de trânsito controlado por dispositivos não tradicionais

Daniel V. Macedo Yvens R. Serpa Ygor R. Serpa Augusto P. Abreu Maria Andréia F. Rodrigues
Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Universidade de Fortaleza (UNIFOR)
Fortaleza-CE, Brasil
{danielvalentemacedo, yvensre, ygor.reboucas, augustodepaula, andreia.formico}@gmail.com

***Resumo*—Neste trabalho, apresentamos IntEducaTrânsito, um jogo 3D interativo e educativo sobre as normas de trânsito, controlado por dispositivos não tradicionais (dois modelos de volantes, *joystick*, *tablet* e *smartphone*), além do teclado. Em particular, tanto o *tablet* quanto o *smartphone* simulam o volante de um carro da auto-escola. No jogo, o motorista deve dirigir pelo ambiente 3D modelado, respeitando os pedestres, os veículos, as placas, as sinalizações horizontais, as sinaleiras e os gestos de autoridade. Testes funcionais comparativos foram conduzidos em vias sujeitas ao cometimento de infrações, usando diferentes dispositivos de controle. Os resultados mostram que IntEducaTrânsito é uma ferramenta lúdica bastante útil e estimulante para o desenvolvimento da atenção, disciplina e autocontrole do jogador. O público-alvo do jogo inclui também crianças e adolescentes que, mesmo não tendo a idade mínima para dirigir e tirar a carteira de habilitação, podem adquirir conhecimentos importantes sobre as normas de trânsito, de forma interativa, enquanto se divertem.**

***Palavras-chave*—jogo 3D interativo; jogo educativo; normas de trânsito; dispositivos de controle não tradicionais**

I. INTRODUÇÃO

Várias ações educativas têm sido realizadas no Brasil pela Coordenadoria de Educação para o Trânsito, vinculada ao Departamento Estadual de Trânsito (DETRAN). O objetivo básico é informar o cidadão sobre as restrições, condições, proibições e obrigações no uso das vias, para a geração de um convívio social mais harmônico no trânsito [1]. Essas ações são direcionadas à conscientização do cidadão e à preservação da vida, sendo consideradas como o meio mais eficaz para a redução de acidentes a médio e longo prazo. As mensagens relativas a essas ações são imperativas e o desrespeito às normas estabelecidas constitui algum tipo de infração [2]. O trabalho de conscientização no trânsito é fundamental e, portanto, qualquer iniciativa nessa direção merece especial atenção.

Nesse contexto, os *serious games* voltados para a educação no trânsito correspondem a uma das formas possíveis e atuais de praticar e difundir os preceitos de segurança e as normas vigentes, bem como as infrações às quais o cidadão está sujeito ao desrespeitar essas regras [1, 2, 3]. Vários são os benefícios dos jogos: melhoria da capacidade de planejamento e organização, maior rapidez (na tomada de decisões e resposta motora) e uma melhor iniciativa do indivíduo diante de problemas.

Na sociedade atual, com freqüência, parte da população não respeita as leis de trânsito em sua totalidade, contribuindo para a geração de um sistema de trânsito problemático/caótico. Os motivos são bastante diversos, mas quase sempre envolvem desinteresse e desinformação dos condutores. Dessa forma, ferramentas gráficas 3D que utilizem recursos computacionais modernos, voltadas para aplicações gráficas na área de visualização interativa e que venham contribuir para a melhoria desse sistema, são bem vindas.

Neste trabalho, apresentamos IntEducaTrânsito, um jogo 3D interativo e educativo sobre as normas de trânsito, controlado por dispositivos não tradicionais (dois modelos de volante, *joystick*, *tablet* e *smartphone*). Em particular, tanto o *tablet* quanto o *smartphone* simulam o volante de um carro da auto-escola. O motorista deve dirigir com cautela o carro da auto-escola pelas vias do ambiente 3D modelado, respeitando os pedestres, os veículos (parados ou em movimento), as placas (de regulamentação, advertência e indicação), as sinalizações horizontais, as sinaleiras e os gestos de autoridade, de tal forma a evitar o cometimento de infrações. Os resultados evidenciam que o IntEducaTrânsito é uma ferramenta lúdica e útil no processo de aprendizado das normas de trânsito, pois gera estímulo, desenvolve a atenção, disciplina e autocontrole. Além disso, viabiliza diferentes experiências aos jogadores: seja na forma de controle, usando diferentes dispositivos; seja interagindo com os próprios objetos e personagens 3D do jogo. O público-alvo do jogo inclui além de adultos, crianças e adolescentes que, mesmo não tendo a idade mínima para dirigir e tirar a carteira de habilitação, podem adquirir conhecimentos importantes sobre as normas de trânsito, de forma interativa, enquanto se divertem.

II. TRABALHOS RELACIONADOS

Existem alguns jogos educativos voltados para a área de trânsito, inclusive, disponíveis em *sites* dos DETRANs de alguns estados do Brasil. Contudo, a maioria deles é do tipo Memória, Erros ou Forca; alguns são mais direcionados ao público infantil [1, 2, 3]. Em geral, estes utilizam o *mouse* e o teclado como dispositivos de entrada e controle do jogo.

Com relação aos métodos para desenvolvimento de jogos, Luz [4] apresenta um esboço de um processo que inclui os elementos gráficos e as etapas que o compõem, similarmente a de um processo de desenvolvimento de *software* como o RUP [5], apropriado para o IntEducaTrânsito e outros tipos de *serious games*. Velasquez [6] apresenta também um modelo de Engenharia de *Software* bastante útil para o desenvolvimento de jogos e simulações interativas, detalhando as fases principais do seu ciclo de vida: coleta e análise de requisitos, projeto, construção, testes e manutenção. O modo como estas fases estão dispostas no ciclo de vida do jogo depende do modelo adotado, por exemplo, clássico (Cascata e Espiral) ou ágil (*SCRUM* [7], *Stage-Gate* [8] e *Extreme Programming* XP [9]).

Já Pereira [10], discorre sobre a capacidade que os jogos eletrônicos têm de envolver crianças, jovens e adultos. O autor afirma que estes são formas alternativas de ensino-aprendizagem, descrevendo que o desenvolvimento de jogos, antes realizado exclusivamente por programadores, atualmente, também pode ser realizado por profissionais não necessariamente atuantes na área de Informática, devido à existência de ferramentas gráficas de mais alto nível que facilitam esse trabalho.

No contexto dos jogos educativos que envolvem trânsito, o jogo *Smart Driver* [11] apresenta a figura de um jogador dirigindo um carro, respeitando algumas das leis existentes. Contudo, este jogo contém um conjunto de leis/infrações bastante restrito, além de não oferecer grandes desafios ao jogador. Basicamente, o motorista deve ficar atento para não ultrapassar a velocidade máxima permitida e evitar acidentes com outros veículos. Adicionalmente, alguns ensinamentos básicos, relacionados às normas de trânsito, são transmitidos ao jogador (uso do cinto de segurança, acomodar as crianças no banco traseiro do veículo, etc.). O jogo foi desenvolvido em *Flash* e é jogado no próprio *browser*.

Outro jogo voltado para a educação no trânsito é o EducaTrans [12], cujo público-alvo são alunos do Ensino Fundamental I-II e Ensino Médio. O jogador tem a opção de escolher o seu papel no jogo: pedestre, ciclista ou motorista. Assim, haverá um conjunto de leis de trânsito que deverá respeitar, de acordo com o papel escolhido. Esse jogo usa um mecanismo de evolução dos pedestres autônomos e oferece um ambiente 3D mais próximo ao de um cenário real, quando comparado a outros trabalhos existentes.

Ainda na linha de *serious game* no trânsito, há também o VRUM [13], cujo público-alvo são crianças e adolescentes. O jogador assume o papel de um jovem de 18 anos que sonha em participar em uma corrida de carros. Para tal, precisa tirar a carteira de habilitação e cumprir diversas missões no jogo. O VRUM foi o primeiro jogo comercial brasileiro voltado para a educação no trânsito.

Além dos jogos educativos, existem também os simuladores de direção que são *softwares,* freqüentemente associados a certos dispositivos de *hardware*, usados na formação de condutores. Esses simuladores já são bastante comuns na Europa e chegam a representar 70% do treinamento de novos motoristas em alguns países [14]. A filosofia dos simuladores é apresentar e antecipar a realidade que o jogador enfrentará quando estiver dirigindo em ruas reais, inclusive, criando situações que estimulem uma reação rápida do aluno. Nessa linha, existe uma iniciativa nacional para incluir no código de trânsito brasileiro a implantação de simuladores nas auto-escolas. Um outro exemplo de simulador de direção bastante popular é o *City Car Driving* [15]. Mais recentemente, em [23] foi implementado um protótipo de um jogo educativo no trânsito bastante simples, que serviu de inspiração inicial para o jogo desenvolvido nesse trabalho.

Backlund *et al*. [16] apresentam um estudo com alunos de auto-escolas. Para identificar diferenças de comportamento entre alunos (com e sem experiência em jogos de computador), os autores deste trabalho focam nas habilidades de direção e segurança do jogador. Reportam que alunos com experiência em jogos têm uma habilidade geral de direção maior do que alunos com pouca experiência ou desprovidos dela.

Outros jogos mais modernos usam o acelerômetro para controle dos objetos virtuais, seleção de itens, movimentos de panorâmica e *zoom* pelo ambiente 3D. Rekimoto foi um dos pioneiros nessa linha [17]. O autor demonstrou que os *designers* podem mapear com sucesso a entrada relativa ao *tilt* angular, de forma discreta ou contínua. Estudos anteriores já haviam evidenciado que, em certas situações, movimentos de *tilting* vertical e rolamento horizontal seria fáceis de serem memorizados e reproduzidos pelo jogador, levando à incidência de um número menor de erros no jogo [18, 19].

III. BLENDER GAME ENGINE

O motor de jogos que escolhemos para o desenvolvimento do IntEducaTrânsito foi o *Blender Game Engine* ou BGE [20], devido à sua robustez, rápida prototipação, por ser gratuito e de código livre. Além disso, o BGE é integrado ao *Blender* [21], uma ferramenta para modelagem geométrica 3D e animação de personagens, o qual é *open source* e multi-plataforma. BGE é escrito em C++ e utiliza a API *OpenGL* para a geração de gráficos, *OpenAL* para o som 3D, o motor de física e detecção de colisões *open source Bullet* e a linguagem *script* orientada a objetos e interpretada *Python* [22] para codificação. Embute, ainda, algumas vantagens do próprio *Blender*: seus modelos são compatíveis com a maioria dos *game engines* existentes no mercado.

A. Blocos de Lógica

Os Blocos de Lógica ou *Logic Bricks* são umas das principais características da estruturação do BGE. Esses elementos são blocos de ferramentas cujas configurações e parâmetros podem ser conectados uns aos outros para definir a lógica e o funcionamento do jogo, podendo ser encontrados no *Blender* em uma área de trabalho chamada *Logic Editor* (ou Editor de Lógica). Desta forma, podemos criar um jogo completo e simples, com diversas funcionalidades, usando apenas as ligações dos Blocos de Lógica. Esta é uma característica bastante interessante para *designers* e modeladores 3D com pouco conhecimento de programação.

Alternativamente, podemos usar codificação via *script Python,* em conjunto com os Blocos de Lógica, os quais dividem-se em três grupos principais (Sensores, Controladores e Atuadores), sendo organizados para que o fluxo da execução

se inicie nos Sensores, passe pelos Controladores e termine nos Atuadores.

Os Sensores são responsáveis pela detecção dos eventos ou os agentes que disparam a execução de uma ação. Há diversos tipos disponíveis:

- Sensor *Always* (ou sempre): instrução direta para acionar um controlador. Pode ser configurado para executar a cada quadro, ou apenas uma única vez;
- Sensores *Mouse*, *Keyboard* e *Joystick*: disparados através do uso de dispositivos de entrada de dados tais como *mouse*, teclado ou controles de jogo; e
- *Collision, Touch* e *Near*: acionados através de colisão, contato ou até mesmo identificação de proximidade entre objetos na cena do jogo.

Basicamente, a função dos Controladores é direcionar a ação para um Atuador ou executar um *script Python*. Funciona através da especificação de elementos que encontramos na lógica condicional, bastante conhecida em programação. Dentre esses elementos, estão as estruturas lógicas *AND* (e) e *OR* (ou).

Os Atuadores são responsáveis pelas ações no jogo, sendo os principais: *Motion* (realiza translação, rotação e escala de objetos), *State* (modifica o estado de um objeto), *Camera* (aplica mudanças nas propriedades da câmera), *Sound* (instrui o *game engine* para executar algum som), *Scene* (possibilita o carregamento de novas cenas), *Visibility* (modifica a visibilidade do objeto associado) e *Action* (reproduz animações).

B. *Propriedades e Estados*

As Propriedades nada mais são do que atributos vinculados aos objetos e servem para armazenar ou salvar informações relacionadas ao objeto. Já um Estado de um objeto corresponde a uma situação na qual um objeto encontra-se em um momento do jogo. Cada Estado possui um conjunto próprio de Blocos de Lógica que atua somente quando este estiver ativo. Existem Blocos de Lógica para gerenciar a mudança de Estado dos objetos.

IV. VISÃO GERAL DO INTEDUCATRÂNSITO

Nas próximas subseções, apresentaremos uma descrição geral dos principais componentes modelados no cenário do jogo e uma visão detalhada dos aspectos de implementação mais relevantes no IntEducaTrânsito.

A. *Modelagem Geométrica e Texturas: Veículos e Ambiente 3D*

No IntEducaTrânsito há 26 veículos, em média, circulando no ambiente 3D que modelamos. Cada veículo possui 1.104 vértices (Fig. 1).

O ambiente ocupa 7.6 MBytes de memória e totaliza 95.067 vértices, contendo uma grande variedade de personagens estáticos e dinâmicos (Fig. 2). Dentre os principais objetos modelados, citamos: a escola (cujo *design* foi inspirado no formato de livros), o *shopping* (de *design* ondulado e com rampas de acesso para pedestres e para carros,

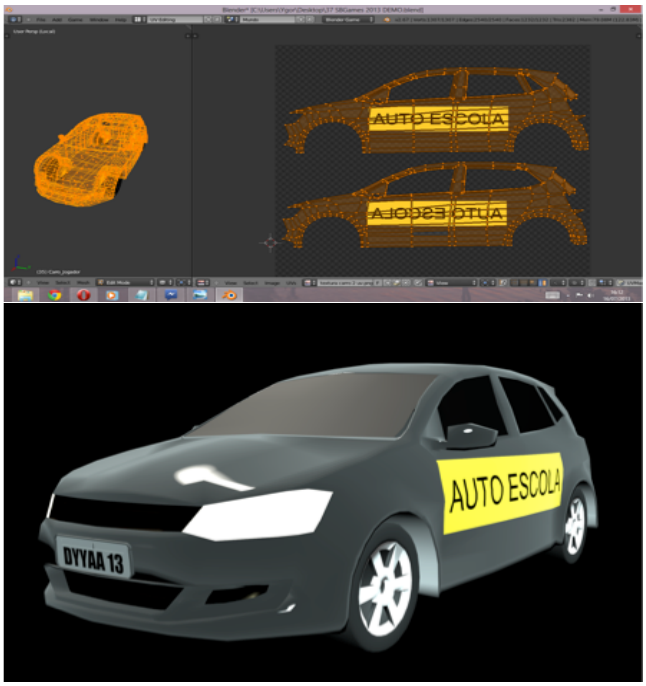

Fig. 1. Modelo 3D do carro da auto-escola.

esta última, gerando um ponto de alto fluxo de veículos na rua correspondente à sua localização, consequentemente, gerando um maior nível de dificuldade no jogo para o motorista do carro da auto-escola), estacionamentos, a auto-escola (ponto de saída do carro do jogador), casas, praças (com bancos, mesas, postes de iluminação, passarelas), jardins, árvores, transeuntes (crianças e adultos), ruas sinalizadas (de sentido único e mão dupla), avenidas com alto fluxo de carros, etc. Contém 44 texturas diferentes, além de modelos de iluminação e tonalização ativados, de acordo com o material do objeto.

B. *Dispositivos de Controle*

Projetamos IntEducaTrânsito para ser controlado também por dispositivos não tradicionais (fora o teclado), tais como: *joystick*, volantes, *tablet* e *smartphone* (Fig. 3). Em particular, os dois últimos dispositivos simulam um volante real via recursos computacionais mais modernos, entre os quais, sensibilidade a toques e movimentos de *tilt*.

O sistema de entrada do usuário é composto por (1) uma Camada de Abstração, responsável pela abstração dos dispositivos; (2) Observadores de Dispositivos, responsáveis pela verificação nos seus estados e notificação de variações existentes nos mesmos; e (3) uma Lógica de Controle, que impede que o jogo seja controlado por vários dispositivos simultaneamente. Para a elaboração deste último componente, estabelecemos uma prioridade entre os dispositivos: volantes, *joystick*, *tablet*, *smartphone* e teclado, nessa ordem. Assim, o dispositivo que terá seus dados escritos na Camada de Abstração será o dispositivo conectado com maior prioridade.

A Camada de Abstração por sua vez é dividida em duas partes: Sensores e Dados. Cada dispositivo possui um sensor responsável pela leitura do estado corrente do dispositivo e pela

alteração de um ou mais dados, se necessário. Dados são informações como, por exemplo, "o freio está acionado", usadas durante o jogo para prover imparcialidade em relação ao dispositivo de entrada ativo. O sensor de teclado apenas lê o estado das teclas e, se preciso, altera os dados correspondentes. O sensor de *joystick*, contudo, é diferente quando se trata dos controladores analógicos, pois geram dados discretos dentro de um intervalo (ao invés de valores *booleanos*) para representar a rotação do volante do veículo.

Os Observadores de Dispositivos observam a ocorrência de possíveis variações no estado destes. Caso essas variações sejam significativas para o jogo, essa informação é transmitida para a Lógica de Controle a qual, por sua vez, escreverá ou não na Camada de Abstração, dependendo da prioridade do dispositivo.

Para cada dispositivo, criamos um Observador associado. O teclado, por exemplo, usa funcionalidades de entrada da própria *game engine*, já os volantes e *joystick* usam uma DLL que escrevemos em C++ para a leitura dos dispositivos (a biblioteca aberta SDL para a leitura dos dados dos dispositivos e os nomes dos mesmos para identificá-los e realizar o mapeamento correto). Os dados coletados são então processados para que as informações de entrada do usuário sejam traduzidas em informações relevantes para o jogo.

No caso do *tablet* e *smartphone*, o sistema que implementamos é relativamente mais complexo do que os demais. Criamos uma aplicação gráfica 3D (usando a *Unity Engine* [24] e a linguagem *C#*) para simular o painel frontal do carro da auto-escola, como uma solução alternativa capaz de prover ao desenvolvedor o reuso de código, possibilitando também a exportação para diferentes plataformas móveis (por exemplo, *iOS* e *Android*). Em particular, essa aplicação contém uma série de controles acessíveis no painel: volante, marcha, pedais (freio e acelerador) e sinaleira, além de outras informações importantes para a condução do veículo: velocímetro, temperatura, nível de gasolina e rotações por minuto do motor (RPM), como mostra a Fig. 4. O jogador pode pressionar tanto o acelerador, quanto o freio. À direita inferior dessa figura, encontram-se os controles da marcha e sinaleira, os quais simulam, respectivamente, uma marcha semi-automática (para aumentá-la deslocamos a alavanca para cima e para reduzi-la, deslocamos para baixo) e as sinalizações do motorista (virar à esquerda ou direita).

No *tablet* e *smartphone*, implementamos o volante do carro usando o sensor de movimento dos dispositivos (acelerômetro), de forma que ao incliná-los para a esquerda ou direita, o volante do carro no IntEducaTrânsito gire de forma correspondente. Os outros controles desses dispositivos foram projetados como botões sensíveis ao toque, posicionados em locais semelhantes às posições convencionais destes em um carro real. Para aumentarmos o nível de imersão, alinhamos o painel interior frontal do carro ao horizonte. Como os computadores tradicionais não têm acelerômetro, desenvolvemos um módulo com o qual, através de uma conexão *Wi-Fi* entre o *tablet/smartphone* e o computador, é possível transmitir dados de inclinação do dispositivo diretamente para o jogo.

Fig. 2. Partes do ambiente 3D modelado no IntEducaTrânsito.

Fig. 3. IntEducaTrânsito sendo controlado por diferentes dispositivos: volantes, *joystick*, *tablet* e *smartphone*.

Além disso, para o *tablet* e o *smartphone*, devido à necessidade de geração de altas taxas de atualização dos dados (em detrimento do risco de perda de pacotes), implementamos um servidor UDP o qual, por padrão, espera uma conexão externa de um dispositivo, ou seja, usamos *sockets* para comunicação via protocolo UDP. Após uma conexão bem sucedida, o jogo captura os dados de entrada enviados pelo dispositivo e, em seguida, transmite a cada quadro os dados relativos ao estado atual do veículo (por exemplo, nível de gasolina e velocidade). A Fig. 5 apresenta um trecho de código do servidor implementado em *Python* no *Blender*. Para que o jogo pudesse "entender" o dispositivo, definimos um protocolo próprio de comunicação capaz de tornar a interface de comunicação via *socket* extensível, propiciando a inclusão de novos dispositivos de entrada.

Uma visão geral da arquitetura do IntEducaTrânsito é mostrada na Fig. 6. Do lado direito, observamos o uso de *sockets*, Unity e C# pelo *tablet* e *smartphone*; do esquerdo, a DLL e C/C++, pelos volantes e *joystick*.

A Lógica de Controle é implementada de uma forma simples, porém, eficaz: cada Observador é executado em seqüência e na ordem de menor para maior prioridade. Conseqüentemente, as alterações realizadas por dispositivos com maior prioridade, sempre sobrescreverão as escritas geradas por dispositivos com menor prioridade.

C. *Sinalizações*

O Código de Trânsito Brasileiro define sinalização de trânsito como um conjunto de sinais de trânsito e dispositivos

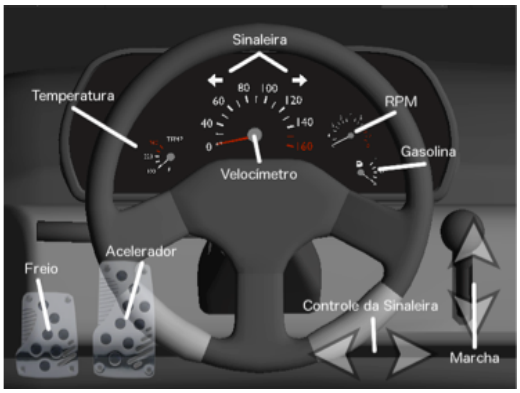

Fig. 4. Aplicação gráfica 3D que simula o painel frontal do carro da auto-escola e seus controles.

de segurança inseridos na via pública para garantir seu uso adequado, possibilitando melhor fluidez no trânsito e maior segurança dos veículos e pedestres que nela circulam [1, 2]. Assim, as sinalizações que modelamos e implementamos no IntEducaTrânsito foram: placas de regulamentação, advertência, sinalização horizontal e sonora (Fig. 7).

Mais especificamente, as placas de regulamentação visam informar os usuários sobre as condições, proibições, obrigações ou restrições no uso das vias, suas mensagens são imperativas e o desrespeito às mesmas constitui infração. Já as de advertência têm como finalidade alertar os usuários sobre condições potencialmente perigosas nas vias, indicando a sua natureza. As sinalizações horizontais são as posicionadas no pavimento das vias para controle, advertência e orientação ou informação. São faixas e marcas feitas no pavimento, com tinta refletiva, sendo as principais marcações nas cores amarela e branca. As sinalizações sonoras são as executadas pelo agente de trânsito (no caso de nosso jogo, pelo guarda em frente à escola, autorizando e impedindo o avanço dos veículos e dos pedestres na passagem sinalizada horizontalmente), usando um apito.

```
import socket
from bge import logic as l

UDP_IP = ""
UDP_PORT = 5006
UDP_PORT_DESTINATION = 5007

cont = l.getCurrentController()
own = cont.owner

if not 'sock' in dir(l):
    l.sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    l.sock.bind((UDP_IP, UDP_PORT))
    l.sock.setblocking(0)

try:
    data, addr = l.sock.recvfrom(1024)
    if data:
        message = str(data, "utf-8")
        words = message.split(' ')
        if words[0] == "PRESS":
            own[words[1]] = True
        elif words[0] == "UNPRESS":
            own[words[1]] = False
        elif words[0] == "SET":
            own[words[1]] = float(words[2])
        elif words[0] == "IP":
            own["UDP_IP_DESTINATION"] = words[1]
except:
    pass
```

Fig. 5. Trecho de código do servidor.

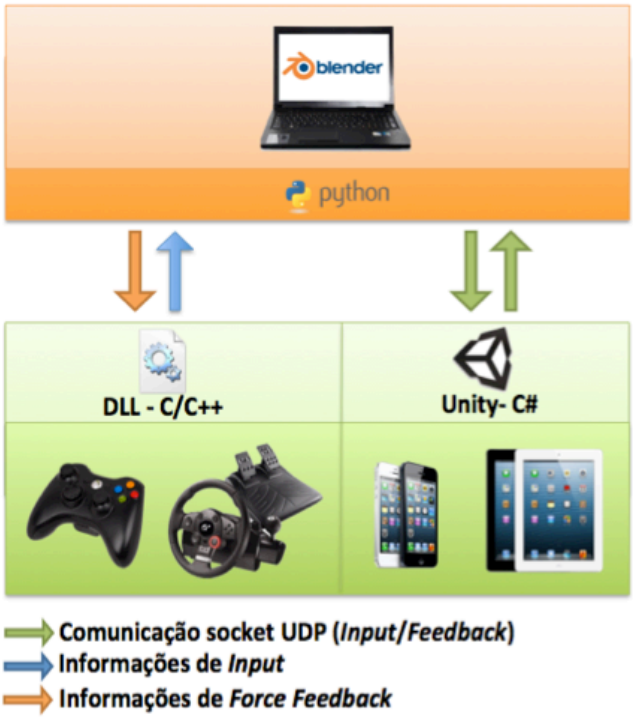

Fig. 6. Arquitetura do IntEducaTrânsito.

Nas ruas do ambiente 3D modeladas existem placas de regulamentação distribuídas com limites de velocidade (30, 40 e 60 km/h) e caixas envoltórias usadas para verificar, durante a colisão com o veículo do jogador, a velocidade do carro, comparando-a com o limite estipulado na via. Caso a velocidade seja maior do que o limite permitido, a infração é computada e mostrada ao jogador no IntEducaTrânsito. Para simplificar o processo de modelagem, a velocidade do veículo é mantida em uma propriedade, chamada "velocidade".

A infração de Proibido Estacionar é detectada quando o carro satisfaz as seguintes condições simultaneamente: está parado, desligado e interceptando uma caixa envoltória marcada para detectar esta infração.

A placa de Siga em Frente só existe nas vias de sentido único. No carro do jogador existem dois objetos vazios (*empties*) nomeados "frente" e "trás". Em cada via que esta infração é detectada há outros dois *empties*, um no começo e outro no final da via, os quais servem para denotar o sentido da mesma (do início para o fim). Quando o carro do jogador estiver contido na caixa envoltória da via, no jogo é verificado se o *empty* "frente" está mais perto do fim da via do que o *empty* "trás". Ou seja, se a frente do carro está mais perto do fim da via do que a sua parte de trás. Caso o *empty* "trás" esteja mais perto do fim do que o "frente", então o carro está transitando do fim para o começo da rua (ou na contramão), conseqüentemente, uma infração é disparada no jogo. Vale ressaltar que este método permite o uso de um mesmo par de *empties* para diversas ruas, contanto que estas tenham o mesmo sentido.

As placas de Sentido Proibido e Proibido Virar à Esquerda/Direita são detectadas por um sistema de referenciais

Fig. 7. Sinalizações implementadas no IntEducaTrânsito.

e caixas envoltórias, estrategicamente posicionadas ao longo do ambiente, para detectar colisão entre o carro da auto-escola e os objetos do ambiente, bem como capturar infrações. O trajeto dos veículos também é formado por caixas envoltórias, invisíveis e irrelevantes para o jogador em termos de colisão com obstáculos.

Em vias de sentido duplo e sem canteiro central (onde são permitidas ultrapassagens pela contramão) não implementamos a detecção de contramão, já que demandaria um nível de complexidade de controle mais detalhado para tratar e distinguir o dirigir na contramão e a realização de uma ultrapassagem.

A infração de Parada Obrigatória é detectada de forma bem simples. Na zona onde o jogador deve parar, criamos uma caixa envoltória. A infração é disparada caso o jogador entre nesta região, com uma velocidade maior que 16km/h. Experimentalmente, verificamos que este limiar propicia uma certa tolerância ao jogador, além de possibilitar que seu veículo não pare completamente.

As placas de advertência, em quase sua totalidade, não possuem implementações diretas de controle lógico associadas a infrações no IntEducaTrânsito, pois são de relevância apenas para o usuário. Contudo, as regras semafóricas devem ser respeitadas, sendo que cruzar o sinal vermelho corresponde à uma infração, bem como avançar a faixa de pedestres (quando houver pedestres atravessando) e desrespeitar as sinalizações do guarda de trânsito.

Implementamos semáforos para controlar o fluxo do trânsito, de forma a bloquear temporariamente o fluxo de uma via para liberar o de outra, alternadamente. Caso o jogador desrespeite o semáforo, isto é, cruze uma via durante o sinal vermelho, o IntEducaTrânsito exibe na tela principal a ocorrência de uma infração. Os semáforos sempre são posicionados em pares no cruzamento das vias, assim, quando um está vermelho (Estado 1) o outro está verde (Estado 0) e

vice-versa. Um dos semáforos do par possui um temporizador que indica a mudança de estado. O diagrama de funcionamento do semáforo é mostrado na Fig. 8.

Outros veículos não controlados pelo jogador (mas pelo jogo) trafegam pelas vias e, ao se depararem com as caixas envoltórias de um semáforo, verificam o seu estado antes de parar ou não o veículo. Caso esteja vermelho e o veículo não proceder com a parada, a infração é sinalizada.

O guarda que modelamos no jogo, sob o ponto de vista da implementação, opera de forma muito similar ao semáforo (Fig. 9). A diferença básica é que o guarda é único (um semáforo somente, com a função de garantir a travessia dos pedestres com sucesso). Visualmente, é um personagem 3D que executa movimentos articulados por intermédio de gestos com os braços e produz sinalização sonora, por meio de um apito.

D. *Indicador, Retrovisor, Suspensão e Sinaleira*

No IntEducaTrânsito há dois tipos de cenas básicas: o ambiente onde estão os carros, as ruas, os transeuntes, as árvores, as sinalizações e as construções em madeira e alvenaria; e o HUD (*Heads Up Display*), onde inserimos um painel com informações sobre o estado atual do carro (motor ligado/desligado, marcha, velocidade, cinto de segurança ativado/desativado e seta superior indicadora da direção de movimento), o painel de notificações das infrações e o retrovisor. Quando o jogo é executado, a cena correspondente ao HUD é incluída sobre a cena do jogo. Implementamos no teto do veículo uma seta vermelha 3D. Esta seta corresponde a um indicador em tempo real, o qual aponta constantemente na direção de um ponto-alvo qualquer, pertencente ao cenário do jogo. Esta seta não possui colisão e é relacionada ao carro da auto-escola pela propriedade *parent* do *Blender*.

O cálculo da rotação do indicador é simples, realizado a partir de um triângulo retângulo. O cateto adjacente ao ângulo de rotação α do veículo corresponde à diferença entre as coordenadas x do indicador e do destino. Analogamente, cálculos similares são feitos para o cateto oposto, porém, com as coordenadas y do indicador e do destino. Com os valores numéricos dos catetos, calculamos o arco-tangente para encontrar o ângulo de rotação no eixo z, em torno do qual será executada a rotação no indicador. Como a posição inicial do indicador do carro aponta para a frente do veículo, subtraímos de $(\pi - \alpha)$ para garantir que a rotação mantenha a ponta da seta voltada para o destino.

Implementamos o retrovisor usando o recurso *video texture* do BGE [20], o qual permite a manipulação de texturas em tempo de execução do jogo. Estas podem ser modificadas a partir de arquivos de vídeo, imagem e renderização de câmeras do jogo. Mais especificamente, o retrovisor que implementamos corresponde a um plano de visão que exibe a imagem capturada por uma câmera posicionada atrás do veículo da auto-escola, com o vetor direção voltado para trás. O *video texture* é aplicado via um *script* associado a um *Sensor Always*, executado continuamente. A textura é capturada, a partir de seu material associado; a fonte de modificação da textura (a câmera posicionada atrás do carro) é definida; e um recurso de *refresh* é continuamente disparado para atualizar a

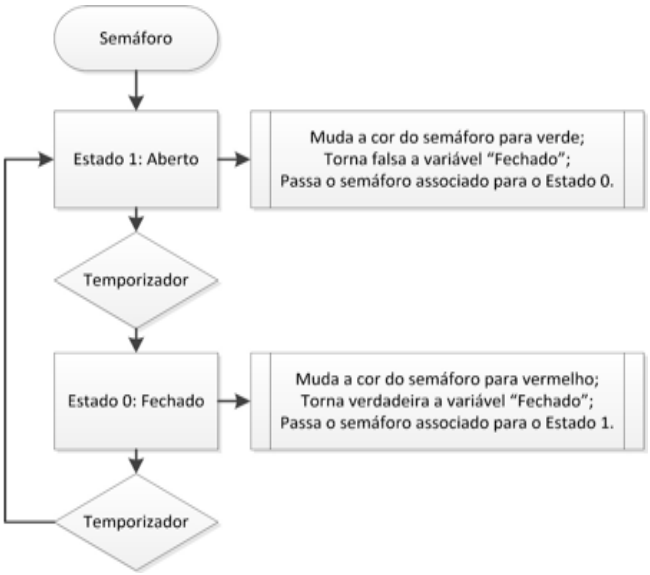

Fig. 8. Diagrama ilustrativo do funcionamento do semáforo.

Fig. 9. O guarda de trânsito realizando, em frente à escola, sinalização gestual e sonora no IntEducaTrânsito.

imagem a ser renderizada nesse plano (Fig. 9, painel superior retangular contido na imagem).

Já o mecanismo de suspensão dos carros do IntEducaTrânsito faz parte da implementação padrão de veículos da biblioteca *Python* do *Blender* (*Vehicle Wrapper*) [22]. Configuramos a mesma através de parâmetros para a montagem do carro e métodos específicos para o ajuste da suspensão: *addWheel* (altura de descanso da suspensão e eixos x, y ou z, em que é disposta; e métodos específicos de configuração (ajuste da compressão, amortecimento e rigidez). A implementação do efeito de suspensão é melhor ilustrada alterando-se os valores desses parâmetros para gerar configurações específicas por rodas e possibilitar a criação de carros com diferentes tipos de suspensão.

A sinaleira é uma ferramenta usada para indicar uma possível conversão à direita ou esquerda. Teoricamente, deve ser acionada com antecedência pelo motorista, sempre que o

veículo for sair de sua trajetória retilínea, isto é, antes de conversões, ultrapassagens ou desvios, atuando como alerta para os demais motoristas sobre as ações planejadas pelo condutor. No IntEducatrânsito, implementamos este recurso somente no veículo do jogador, sendo considerada infração, qualquer conversão sem o correto acionamento da sinaleira. No jogo, a fim de promover uma maior semelhança entre um volante real e os dispositivos, optamos por mapear certas funcionalidades, como a sinaleira de um volante real, para certos botões dos dispositivos com características lógicas e espaciais próximas às de um volante real. Por exemplo, a sinaleira (esquerda e direita, respectivamente) é acionada pelos seguintes controles: botões 4 (L1) e 5 (R1) do *joystick*; botões 1 e 3 do teclado; botões L1 e R1 do volante simples; "orelhas" esquerda e direita dos volantes; e setas horizontais no *tablet* e *smartphone,* acionadas via toque na tela (as quais disparam uma luz no painel, similar ao piscar de uma sinaleira de um carro real).

Na implementação da sinaleira, definimos os valores -1, 0 e 1 para representar os estados "esquerda", "desligada" e "direita", respectivamente. A lógica para que as sinaleiras esquerda e direita não sejam ativadas simultaneamente, reside nos Observadores de Dispositivo. Ao acionar o botão relativo a uma delas, geramos um efeito visual de "pisca-pisca" no material do objeto. Realizamos então uma varredura pelos materiais do veículo para buscar uma referência ao objeto do material da sinaleira. Encontrado o objeto do material, um *looping* é iniciado, alternando a sua cor entre amarelo claro e escuro, gerando o efeito "pisca-pisca". Quando uma das sinaleiras é desligada, a cor retorna para *default*, isto é, amarelo escuro.

No jogo também incluímos efeitos sonoros a partir de um *script Python* e *Logic Bricks* no BGE, os quais também foram adicionados ao painel do veículo no *tablet* e *smartphone*: barulho do motor, que aumenta de acordo com o aumento da velocidade; som da sinaleira ao piscar; e apito do guarda.

V. RESULTADOS E CONCLUSÕES

IntEducaTrânsito é um jogo educativo atual e bastante divertido para crianças, jovens e adultos e potencialmente útil como ferramenta alternativa de auxílio ao processo de habilitação em centros de formação de condutores. O jogo oferece ainda várias possibilidades de extensão, tais como, a inclusão de formas de recompensa, avaliação, compartilhamento de *ranking* de pontuações em redes sociais, etc. Apesar das infrações serem disparadas com sucesso e em tempo real durante o jogo, dedicamos pouca atenção à criação de premiações ou metas a cumprir.

Sob o ponto de vista da visualização interativa, IntEducaTrânsito apresenta um *design* próprio, com gráficos 3D otimizados e acabamento realista (incluindo texturas, modelos de iluminação e de tonalização), bem como animações de personagens articulados (transeuntes).

À medida que o realismo gráfico e de animação, bem como o nível de interatividade do jogo foi aumentando, percebemos uma maior satisfação e interesse dos jogadores, porém, em plataformas menos robustas, o programa do jogo no BGE apresentou uma certa perda de desempenho, mantendo no geral uma taxa mínima de 40 quadros/s. Provavelmente, isso ocorreu devido ao maior detalhamento da física e lógica de controle e pelo *Python*, que apesar de ser uma ferramenta dinâmica e versátil, ofereceu pouca flexibilidade para a construção de código, gerando erros muitas vezes difíceis de serem diagnosticados, por exemplo, em trechos com diferentes indentações no *script*. Observamos também que o BGE dificulta, em certos momentos, a depuração de variáveis e objetos dinamicamente criados.

Um outro aspecto a ser melhorado no BGE diz respeito ao uso obrigatório de *Logic Bricks* (mesmo em objetos que usam *script Python* para tarefas simples, como emitir um som), não sendo a solução ideal para todos os eventos do jogo. Uma possibilidade melhor seria criar um *workflow* no qual fosse possível usar somente *Python* para a lógica do jogo.

Com relação ao uso de diferentes dispositivos, estudos preliminares indicaram que controlar o IntEducaTrânsito com um *joystick* ou teclado agrega pouca novidade, desafio ou estímulo aos jogadores, particularmente, nos movimentos de "passar a marcha". Por outro lado, o uso dos volantes aumentou notadamente o nível de imersão, desafio e diversão dos jogadores. Para usuários menos acostumados com o *tablet* e *smartphone,* detectamos certa perda de controle do carro, principalmente, por serem dispositivos leves e controlados por comandos *touch screen* e propiciarem ampla liberdade de movimentos. Já usuários mais familiares com o uso destes dispositivos mostraram um maior controle no jogo.

Como trabalhos futuros podemos citar a inclusão de metas a cumprir e de recompensas no jogo, bem como a realização de avaliações comparativas detalhadas sobre a jogabilidade do IntEducaTrânsito, usando todos os dispositivos que ele atualmente suporta (teclado, *joystick*, volantes, *tablet* e *smartphone*), a ser calculada empiricamente a partir da velocidade e precisão do usuário em completar um conjunto de tarefas.

AGRADECIMENTOS

Daniel Valente Macedo gostaria de agradecer à CAPES, Yvens Rebouças Serpa e Maria Andréia Formico Rodrigues ao CNPq (Processos No. 157.257/2012-6 e No. 310434/2010-6, respectivamente) e Ygor Rebouças Serpa à FUNCAP-CE (Processo No. 0074 - 00006.01.40/13), pelo apoio financeiro recebido.

REFERÊNCIAS

[1] DETRAN-PR. Jogo de erros. Disponível em: <*educacaotransito.pr.gov.br/anima.html*>.

[2] DETRAN-SE. Jogo de forca. Disponível em: <*www.detran.se.gov.br/jogos_forca.asp*>.

[3] Smart Kids. Jogo de memória. Disponível em: <*www.smartkids.com.br/jogos-educativos/jogo-da-memoria-transito.html*>.

[4] M. H. Luz, Desenvolvimento de jogos de computador. Monografia de Graduação. Departamento de Matemática e Computação, Universidade Federal de Itajubá, 2004.

[5] IBM. RUP (*Rational Unified Process*). Disponível em: <*www-01.ibm.com/software/awdtools/rup*>.

[6] C. E. Velasquez, Modelo de engenharia de *software* para o desenvolvimento de jogos e simulações interactivas. Monografia de Graduação. Universidade Fernando Pessoa, Porto, 2009.

[7] K. Schwaber e J. Sutherland, SCRUM guides. Disponível em:

<pt.wikipedia.org/wiki/Scrum>.
[8] Stage-Gate International. Stage-Gate agile development. Disponível em: *<www.stage-gate.com/knowledge_stage-gate_agile.php>*.
[9] Extreme Programming (XP). Disponível em: *<en.wikipedia.org/wiki/Extreme_programming>*.
[10] A. M. Pereira, "Desenvolvimento de jogos digitais como estratégia de aprendizagem", Informática Educativa II, Fascículo do NEAD/CREAD/UFES, Vitória, ES, 2003.
[11] Smart Driver. Disponível em: *<www.joguegratis.com.br/jogos-educativos/smart-driver>*.
[12] G. A. de Assis, I. K. Ficheman, A. G. D. Corrêa, M. Lobo Netto, R. D. Lopes, "EducaTrans: um Jogo educativo para o aprendizado do trânsito", Revista Renote - Novas Tecnologias na Educação, CINTED, vol. 4(2), UFRGS, Dezembro, 2006.
[13] Thinkbox Games. VRUM. Disponível em: *<www.jogovrum.com.br>*. Acesso em: 05/05/2013.
[14] Olhar Digital. Simuladores de trânsito. Disponível em: *<olhardigital.uol.com.br/produtos/central_de_videos/simuladores>*.
[15] City Car Driving. Disponível em: *<citycardriving.com>*.
[16] P. Backlund, H. Engstrom, M. Johannsesson, "Computer gamming and driving education", Suécia, 2006.
[17] J. Rekimoto, "Tilting operations for small screen interfaces", In Proceedings of the 1996 UIST, pp. 167-168, 1996.
[18] A. Crossan, J. Williamson, S. Brewster e R. Murray-Smith, "Wrist rotation for interaction in mobile contexts", In Proceedings of the 2008 Mobile HCI, pp. 435-438, 2008.
[19] M. Rahman, S. Gustafson, P. Irani, S. Subramanian, "Tilt techniques: investigating the dexterity of wrist-based input", In Proceedings of the 2009 CHI, pp. 1943-1952.
[20] Blender.org. BGE (*Blender Game Engine)*. Disponível em: *<wiki.blender.org/index.php/Doc:2.6/Manual/Game_Engine>*.
[21] Blender.org. *Blender*. Disponível em: *<www.blender.org>*.
[22] Python.org. *Python programming language*. Disponível em: *<www.python.org>*.
[23] A. P. Abreu, "EducaTrânsito: um protótipo de um jogo educativo utilizando o *Blender Game Engine*". Monografia de Graduação. Curso de Ciência da Computação, Universidade de Fortaleza (UNIFOR), junho 2013.
[24] Unity Technologies. Disponível em: *<unity3d.com>*.