CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC	
(X) PRÉ-PROJETO ( ) PROJETO	ANO/SEMESTRE: 2017/2

TOWELJS: ENGINE 3D EM JAVASCRIPT USANDO ARQUITETURA BASEADA EM COMPONENTES.

Gabriel Zanluca

Dalton Solano dos Reis

1 INTRODUÇÃO

De acordo com uma pesquisa realizada pela Newzoo (2017) em 2017 a previsão é que a indústria de jogos eletrônicos movimente 108.9 bilhões de dólares, isso representa um ganho de 7.8 bilhões comparado com o ano de 2016. Dessa forma a indústria de jogos eletrônicos se demostra como algo atrativo para se investir em desenvolvimento visto que as projeções para os próximos anos também indicam aumento.

Com todo o crescimento de desenvolvimento de jogos eletrônicos e comum que apareçam ferramentas que possam facilitar e acelerar a sua construção, e dentre elas temos o uso de motores de jogos. Os motores de jogos facilitam a vida dos desenvolvedores porque neles já existem rotinas, básicas relacionadas com a construção de aplicações gráficas e testadas que garante estarem funcionando, abstraem parte do que precisa ser feito para ter-se elementos comuns em jogos (desenho de objeto, leituras de periféricos, transformações geométricas e em alguns casos até tratamento de física). Os motores também garantem a economia de tempo por não precisar refazer sempre as mesmas rotinas do zero, e como já foi dito por terem já sido testadas garante menos erro ao usa-las. Assim dessa forma o foco fica na jogabilidade do usuário e na história que o desenvolvedor quer contar.

Pensando-se na parte do desenvolvimento de um motor de jogos uma das opções a se levar em consideração seria o uso de uma arquitetura baseada em componentes, que se mostra interessante pelo fato de ser "[...] caracterizado pela composição de partes já existentes, ou pela composição de partes desenvolvidas independentemente e que são integradas para atingir o objetivo final [...]" (FEIJÓ, 2007, p. 17). Trazendo isso para o mundo do desenvolvimento de jogos o benefício seria o fato de comportamentos comuns das personagens serem implementados como componentes e assim serem adicionados sempre que preciso. Por exemplo, um comportamento de pulo poderia ser implementado como um componente de pulo e sempre adicionando quando as personagens que necessitem dele.

Visto todos os argumentos citados a cima o trabalho proposto visa desenvolver um motor de jogos que auxilie o desenvolvimento de jogos em 3D utilizando a linguagem JavaScripts e arquitetura baseada em componentes.

1.1 OBJETIVOS

O objetivo é desenvolver um motor de jogos 3D utilizando arquitetura baseada em componentes para facilitar o desenvolvimento de jogos em JavaScript.

Os objetivos específicos são:

a) criar um ambiente mínimo para renderização em 3D contendo um renderizador, estrutura para criar uma cena, iluminação e uma câmera;
b) disponibilizar a renderização de objetos gráficos básicos como cubos e esferas;
c) desenvolver componentes dedicados para análise da performance.

2 TRABALHOS CORRELATOS

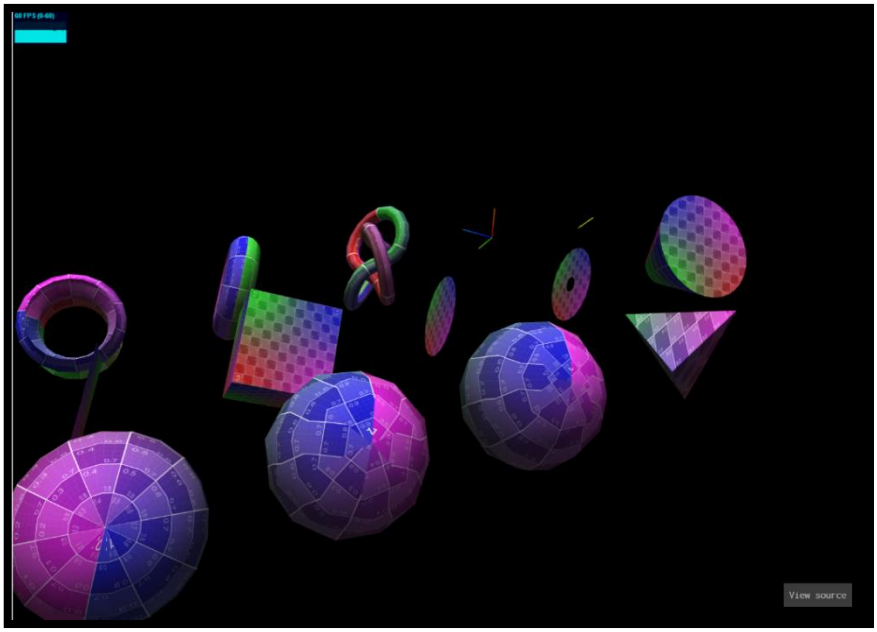
São apresentados trabalhos com características semelhantes aos principais objetivos do estudo proposto. O primeiro é a biblioteca Three.js (THREEJS, 2017), segundo é o editor WebGLStudio.js (WEBGLSTUDIOJS, 2017) e o terceiro é o motor de jogos VisEdu-Engine (HARBS, 2013).

2.1 THREE.JS

O objetivo da Three.js é "[...] criar uma biblioteca 3D leve e fácil de usar. A biblioteca fornece renderizadores <canvas>, <svg>, CSS3D e WebGL" (THREEJS, 2017a, tradução nossa). Desenvolvida em JavaScript e podendo ser vista como uma camada acima se comparado com uso do WebGL puro, porque permite um nível maior de abstração na questão gráfica do desenvolvimento da aplicação. Sendo gratuita para uso e de código aberto disponível para alteração.

Dentre algumas características presentes pode-se listar a presença de formas básicas já definidas tais como: cubos, quadrados, círculos, cilindros, esferas, linhas, etc. Também já conta com elementos de luz, diferentes tipos de matérias além de permitir a adição de textura nos objetos e permite o carregamento de objetos externos produzidos em programas de terceiros. Na Figura 1 pode-se ver um exemplo da biblioteca com algumas das formas geométricas disponíveis e ainda com o uso da iluminação.

Figura 1 – Exemplo do uso Three.js (geometries)



Fonte: Three.js (2017b).

Além dos recursos já citados a biblioteca conta também com recursos para ajudar na criação de animação de objetos segundo Three.js (2017c, tradução nossa) “[...] você pode animar várias propriedades de seus modelos: *bones* de um modelo *skinned and rigged*, *morph targets*, propriedades diferentes do material (cores, opacidade, valores lógicos), visibilidade e transformações”. Além dessa facilidade na hora de criar-se animações questões mais simples como transformações geométricas também são mais fáceis de se fazer e exigem menos linhas de código em comparação ao uso do WebGL.

Por fim caso o programador queira ter tratamento de física na Three.js precisará utilizar biblioteca de terceiros, porque ela não fornece esse recurso, visto que seu foco é a parte gráfica da aplicação.

2.2 WEBGLSTUDIO.JS

O WebGLStudio.js é um editor de gráficos 3D que pode ser acessado totalmente num navegador para se criar os projetos, de código aberto e usa internamente como biblioteca gráfica a LiteScene (LITESCENE, 2017) e tanto o WebGLStudio.js quanto a LiteScene são implementados em JavaScript.

Tendo como principais características (WEBGLSTUDIOJS, 2017, tradução nossa):

- motor de gráficos 3D completo (LiteScene.js) que suporta múltiplas luzes, shadowmaps, reflexões em tempo real, materiais personalizados, postFX, skinning, animação e muito mais;
- sistema baseado em componentes fácil de expandir (controlando a linha de renderização ou encaixar eventos para de interação);
- editor WYSIWYG fácil de usar, com todos os recursos em um só lugar (codificação, composição gráfica e linha de tempo);
- editor de gráficos para criar comportamentos interessantes ou efeitos de pós-produção;
- sistema virtual de arquivos para armazenar todos os recursos na web LiteFileSystem.js apenas arrastando-os (com cotas, usuários e pastas compartilhadas);
- fácil de exportar e compartilhar, apenas enviando um link.

Na Figura 2 pode se observar o uso do WebGLStudio.js com uma cena já carregada com um objeto 3D modelado, e a presença de luz em cena. A direita podemos ver o grafo de cena montado de forma visual, além disso também há campos para configurar as propriedades do objeto selecionado.

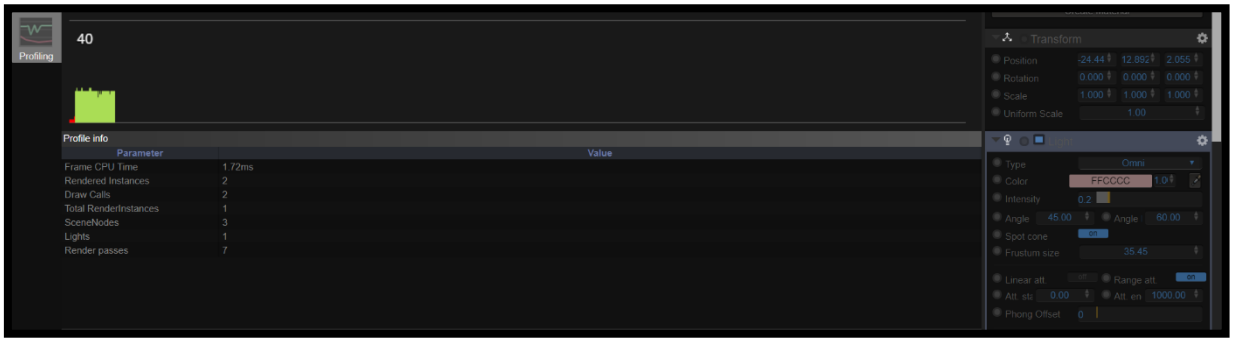
Figura 2 – Uso do WebGLStudio.js

Fonte: WebGLStudio.js (2017).

Semelhante à Three.js o WebGLStudio.js não conta com recursos para tratar questões relacionadas a física, porém conta com uma aba no editor chamada *Profiling* que pode ser vista

na Figura 3 onde é exibido algumas informações relativas a cena atual. E também conta com questões para criar as animações de personagens, podendo até se fazer uso da interface gráfica para configurar alguns dos parâmetros.

Figura 3 – Informações na aba *Profiling*



Fonte: WebGLStudio.js (2017).

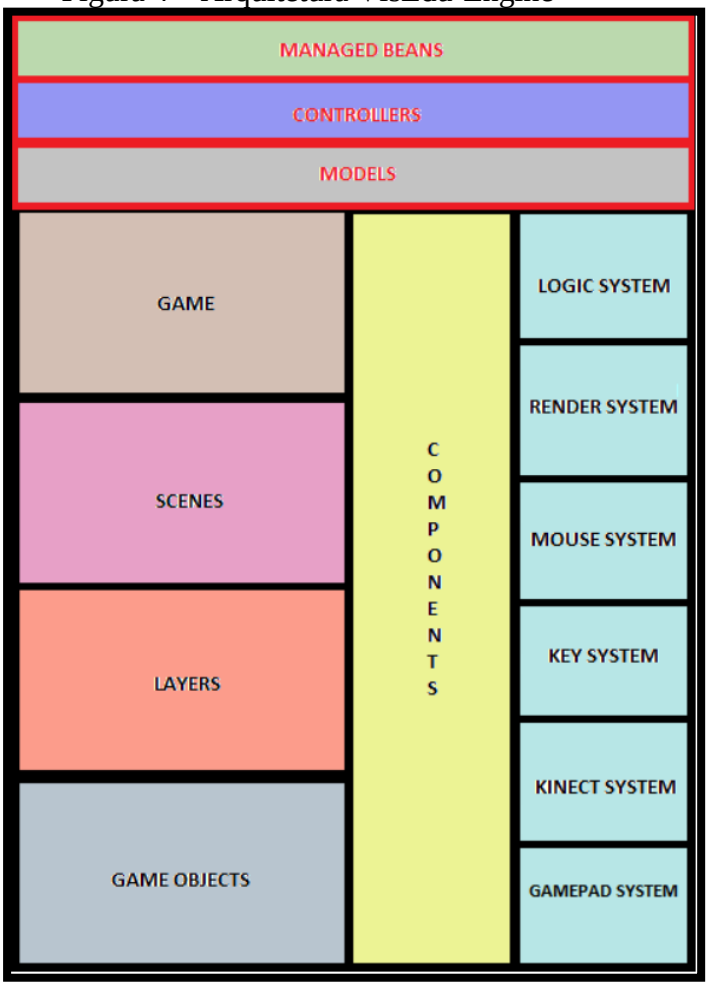
2.3 VISEDU-ENGINE

O VisEdu-Engine é um motor de jogos 2D com as seguintes funcionalidades "[...] criação de cena, criação de camadas, gerenciamento de objetos, gerenciamento de recurso de imagens e áudio, detecção de colisão, física de corpos rígidos (utilizando a biblioteca Box2DJS) e uma arquitetura reutilizável orientada a componentes" (HARBS, 2013).

Segundo Harbs (2013) a arquitetura do motor de jogos foi especificada de maneira orientada a componentes. Na Figura 4 pode se observar o diagrama da arquitetura do VisEdu-Engine onde a comunicação das camadas Game, Scenes, Layers e Game Objects, ocorre no sentido de cima para baixo, ou seja, o Game se comunica com Scenes que se comunica com Layers que por sua vez se comunica com Game Objects. Já todas as camadas com system no nome têm como objetivo criar interface com dispositivos externos e essas por sua vez se comunicam com as camadas citadas anteriormente por meio de componentes.

Como também pode ser observado além do motor ter suporta a periféricos comuns como *mouse* e teclado, há também suporte para o Kinect e *joystick*.

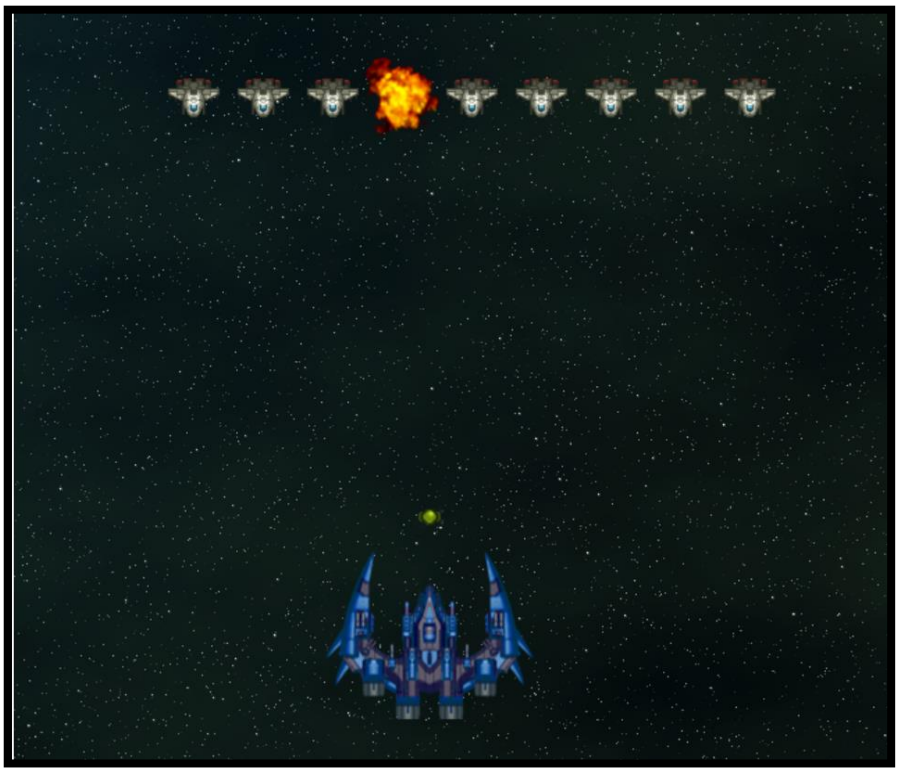
Figura 4 – Arquitetura VisEdu-Engine



Fonte: Harbs (2017).

Os objetos disponíveis no VisEdu-Engine são quadrados, círculos, e polígonos, todos com seu respectivo corpo rígido também para que possa ocorrer o tratamento de colisões. Em questão de organização a aplicação consiste em criar uma cena com várias camadas (*layers*), para que assim tenha-se a impressão de algo estar no fundo ou mais à frente. Na Figura 5 pode-se observar a criação de uma aplicação inspirada no jogo Space in Invaders utilizando o motor de jogos.

Figura 5 – Utilização do motor de jogos VisEdu-Engine



Fonte: Harbs (2017).

3 PROPOSTA DO MOTOR DE JOGOS

Este capítulo tem como objetivo apresentar a justificativa do trabalho proposto, tais como seus requisitos e metodologias de desenvolvimento para elaboração do mesmo.

3.1 JUSTIFICATIVA

No Quadro 1 pode ser visto uma comparação das características dos três trabalhos correlatos apresentados anteriormente.

Quadro 1 – Comparativo entre os trabalhos correlatos

Características	Three.js	WebGLStudio.js	VisEdu-Engine
implementado em JavaScript	Sim	Sim	Sim
possuir grafo de cena	Sim	Sim	Não
sistema baseado em componentes	Não	Sim	Sim
gráfico em 3D	Sim	Sim	Não
possui um editor	Sim	Sim	Sim

Como pode ser visto todos os três trabalhos foram implementados em JavaScript e possuem um editor. Tanto a Three.js quando o WebGLStudio.js possuem a opção de se trabalhar com gráficos em 3D e 2D (fixando a câmera em um dos eixos) já o VisEdu-Engine possui apenas a opção em 2D.

Considerando outras características relativas a computação gráfica a estrutura conhecida como grafo de cena só não é suporta no VisEdu-Engine, visto que essa é uma característica importante, principalmente quando se quer fazer alguma transformação em vários objetos ao mesmo tempo. Todos exceto a Three.js usam uma arquitetura baseada em componentes.

Visto essas características o trabalho proposto apresenta relevância pelo fato de ser um motor de jogos que contemplará todas as características apresentadas exceto possuir um editor e em adicional conterá componentes específicos para monitoramento de performance. Além disso ao final do estudo pretende se ter uma espécie de guia para auxiliar no desenvolvimento de motores de jogos e também ter uma compreensão maior do uso do WebGL e do seu pipeline gráfica.

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O motor de jogos proposto nesse trabalho deverá:

a) O motor de jogos deverá permitir a criação/renderização de objetos gráficos (Requesito Funcional - RF);
b) O motor de jogos deverá permitir a criação de novos componentes (RF);
c) O motor de jogos deverá implementar a estrutura de grafo de cena (RF);
d) O motor de jogos deverá implementar a seleção de objetos utilizando algoritmo baseado em *raycast* (RF);
e) O motor de jogos deverá ser implementado utilizando JavaScript na versão 6 (Requisito Não Funcional - RNF);
f) O motor de jogos deverá contar com componentes próprios para fazer análise da performance (RNF);
g) O motor de jogos deverá ser implementado utilizando a arquitetura baseada em componentes (RNF).

3.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) levantamento bibliográfico: buscar fontes bibliográficas sobre o desenvolvimento de motor de jogos, computação gráfica, WebGL e arquitetura baseada em

componentes;

b) elicitação de requisitos: nessa etapa os requisitos serão reavaliados e se necessário alterados, pode ocorrer também a inclusão caso isso seja percebido na etapa anterior;
c) especificação: utilizar a Unified Modeling Language (UML) para elaborar os diagramas de casos de uso e de classes;
d) implementação: implementar o motor de jogos com base nos requisitos levantados no item (b) e com a modelagem realiza no item (c);
e) documentação: documentar as funções métodos presente no código desenvolvido para melhor compreensão caso haja continuação na implementação por outra pessoa;
f) teste: elaborar testes para validar o motor de jogos implementados, focando no desempenho do mesmo. Além disso criação de cenários para que se ocorra os testes.

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro 2 - Cronograma

	2018									
	fev.		mar.		abr.		maio		jun.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2
levantamento bibliográfico										
elicitação de requisitos										
especificação										
implementação										
documentação										
teste										

Fonte: elaborado pelo autor.

4 REVISÃO BIBLIOGRÁFICA

Este capítulo tem como objetivo descreve brevemente os assuntos que fundamentarão o trabalho a ser desenvolvido dentre eles: motor de jogos e arquitetura baseada em componentes.

Como uma das ideias principais da engenharia de software é evitar o retrabalho no assunto programação o desenvolvimento de jogos não ficou de fora. Segundo a ideia principal de um motor de jogos “[...] é permitir que os recursos comuns a quase todos os jogos sejam reutilizados para cada novo jogo criado. Neste caso, a cada novo jogo, se implementa apenas seus requisitos particulares” (PESSOA, 2001).

A arquitetura baseada em componentes baseia-se no desenvolvimento usando componentes que segundo Sametinger (1997, p.2, tradução nossa) componentes são artefatos que nós claramente identificamos em nossos sistemas de software. Eles têm uma interface, encapsulamento interno detalhado e são documentados separadamente.

REFERÊNCIAS

FEIJÓ, R. H. B. **Uma arquitetura de software baseada em componentes para visualização de informações industriais**. 2007. 88 f. Dissertação (Mestrado em Ciências) - Curso de Pós-graduação em Engenharia Elétrica, Universidade Federal de Rio Grande do Norte, Natal.

HARBS, Marcos. **Motor para Jogos 2D Utilizando HTML5**. 2013. 78 f. Trabalho de Conclusão de curso (Bacharelado em Ciência da Computação) – Centro de Ciências e Exatas Naturas, Universidade Regional de Blumenau, Blumenau.

NEWZOO. **The Global Games Market Will Reach $108.9 Billion In 2017 With Mobile Taking 42%** [S.l.], 2017 Disponível em: https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42. Acesso em: 9 set. 2017.

LITESCENE. **litescene.js**. [S.l.], 2017a. Disponível em: < https://github.com/jagenjo/litescene.js>. Acesso em: 13 set. 2017.

PESSOA, Carlos A. C. **wGEM: um framework de desenvolvimento de jogos para dispositivos móveis**. 2001. Dissertação (Mestrado) — UFPE.

THREE.JS. **three.js**. [S.l.], 2017a. Disponível em: < https://github.com/mrdoob/three.js/>. Acesso em: 9 set. 2017.

THREE.JS. **three.js / examples**. [S.l.], 2017b. Disponível em: < https://threejs.org/examples/#webgl_geometries>. Acesso em: 9 set. 2017.

THREE.JS. **three.js docs - Animation System**. [S.l.], 2017c. Disponível em: < https://threejs.org/docs/index.html#manual/introduction/Animation-system>. Acesso em: 13 set. 2017.

SAMETINGER, Johannes. **Software Engineering with Reusable Components**. New York, Springer, 1997.

WEBGLSTUDIOJS. **webglstudio.js**. [S.l.], 2017. Disponível em: < https://github.com/jagenjo/webglstudio.js/>. Acesso em: 9 set. 2017.

ASSINATURAS

(Atenção: todas as folhas devem estar rubricadas)

Assinatura do(a) Aluno(a): _____

Assinatura do(a) Orientador(a): _____

Assinatura do(a) Coorientador(a) (se houver): _____

FORMULÁRIO DE AVALIAÇÃO (PRÉ-PROJETO – PROFESSOR DE TCC)

Acadêmico(a): _____

Avaliador(a): _____

	ASPECTOS AVALIADOS[1]	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?			
	O problema está claramente formulado?			
	2. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?			
	Os objetivos específicos são coerentes com o objetivo principal?			
	3. TRABALHOS CORRELATOS: São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?			
	4. JUSTIFICATIVA: Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?			
	São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?			
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?			
	5. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?			
	6. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?			
	Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?			
	7. LEVANTAMENTO E FONTES BIBLIOGRÁFICAS Os assuntos relacionados são suficientes e têm relação com o tema do TCC?			
	As fontes bibliográficas indicadas contemplam adequadamente os assuntos relacionados (são indicadas obras atualizadas e as mais importantes da área)?			
ASPECTOS METODOLÓGICOS	8. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?			
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?			
	9. ORGANIZAÇÃO E APRESENTAÇÃO GRÁFICA DO TEXTO A organização e apresentação dos capítulos, seções, subseções e parágrafos estão de acordo com o modelo estabelecido?			
	10. ILUSTRAÇÕES (figuras, quadros, tabelas) As ilustrações são legíveis e obedecem às normas da ABNT?			
	11. REFERÊNCIAS E CITAÇÕES As referências obedecem às normas da ABNT?			
	As citações obedecem às normas da ABNT?			
	Todos os documentos citados foram referenciados e vice-versa, isto é, as citações e referências são consistentes?			

Observações: _____

_____

Assinatura: _____ Data: _____

[1] Quando o avaliador marcar algum item como atende parcialmente ou não atende, deve obrigatoriamente indicar os motivos no texto, para que o aluno saiba o porquê da avaliação.

FORMULÁRIO DE AVALIAÇÃO (PRÉ-PROJETO – PROFESSOR AVALIADOR)

Acadêmico(a): _____

Avaliador(a): _____

	ASPECTOS AVALIADOS[1]	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?			
	O problema está claramente formulado?			
	2. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?			
	Os objetivos específicos são coerentes com o objetivo principal?			
	3. TRABALHOS CORRELATOS: São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?			
	4. JUSTIFICATIVA: Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?			
	São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?			
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?			
	5. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?			
	6. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?			
	Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?			
	7. LEVANTAMENTO E FONTES BIBLIOGRÁFICAS Os assuntos relacionados são suficientes e têm relação com o tema do TCC?			
	As fontes bibliográficas indicadas contemplam adequadamente os assuntos relacionados (são indicadas obras atualizadas e as mais importantes da área)?			
ASPECTOS METODOLÓGICOS	8. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?			
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?			

Observações: _____

_____

Assinatura: _____ Data: _____

[1] Quando o avaliador marcar algum item como atende parcialmente ou não atende, deve obrigatoriamente indicar os motivos no texto, para que o aluno saiba o porquê da avaliação.