UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO DE ENGENHARIA QUÍMICA
CURSO DE ENGENHARIA QUÍMICA

DIEGO DANIEL RODRIGUES INSAURRALDE

SÍNTESE E DESENVOLVIMENTO DE UM SIMULADOR COM REALIDADE VIRTUAL (VR) DE BIODIGESTÃO PARA PRODUÇÃO DE BIOGÁS

TRABALHO DE CONCLUSÃO DE CURSO

Londrina
2023

DIEGO DANIEL RODRIGUES INSAURRALDE

SÍNTESE E DESENVOLVIMENTO DE UM SIMULADOR COM REALIDADE VIRTUAL (VR) DE BIODIGESTÃO PARA PRODUÇÃO DE BIOGÁS

SYNTHESIS AND DEVELOPMENT OF A VIRTUAL REALITY (VR) ANAEROBIC DIGESTION SIMULATOR FOR BIOGAS PRODUCTION

Trabalho de Conclusão de Curso de graduação, apresentado à Universidade Tecnológica Federal do Paraná (UTFPR), Campus Londrina como requisito parcial para obtenção do título de Engenheiro Químico.
Orientador: Prof. Dr. Lucas Bonfim Rocha

Londrina

2023

DIEGO DANIEL RODRIGUES INSAURRALDE

SÍNTESE E DESENVOLVIMENTO DE UM SIMULADOR COM REALIDADE VIRTUAL (VR) DE BIODIGESTÃO PARA PRODUÇÃO DE BIOGÁS

Trabalho de Conclusão de Curso de graduação, apresentado à Universidade Tecnológica Federal do Paraná (UTFPR), Campus Londrina como requisito parcial para obtenção do título de Engenheiro Químico.
Orientador: Prof. Dr. Lucas Bonfim Rocha

Data de aprovação: 15 de Dezembro de 2023

Lucas Bonfim Rocha

Doutorado em Engenharia Química Universidade Estadual de Maringá

Felipi Luiz de Assunção Bezerra

Doutorado em Engenharia Química Universidade Estadual de Campinas

---

Thiago Leandro De Souza

Doutorado em Ciências Universidade Estadual de Maringá

Londrina

2023

Dedico o presente trabalho ao meu pai, José, que esteve comigo no primeiro dia, você é minha fonte de inspiração.

Dedico à minha esposa, Rafaela, presente em minha vida em todos os momentos, sendo o amor da minha vida.

Dedico à minha filha Luísa, que me inspira ser uma pessoa melhor.

Dedico à minha mãe, por toda dedicação que teve comigo.

AGRADECIMENTOS

Primeiro, agradeço a Deus.

Agradeço a todos os discentes que contribuíram para a evolução dos meus conhecimentos ao longo de toda a graduação. Em especial, expresso minha profunda gratidão ao meu orientador, Lucas, pelo grande apoio e incentivo na construção deste trabalho, pelos quais serei eternamente grato.

À minha esposa, agradeço por todo carinho, apoio e amor incansáveis, com você tudo é mais fácil. À minha filha, Luísa, que, mesmo em tão pouco tempo de vida, causou um impacto profundo em mim.

À minha mãe, Elisa, que se dedicou incansavelmente ao meu sucesso desde o meu nascimento. Ao meu pai, que esteve comigo desde o primeiro dia. Nunca vou esquecer a sensação de compartilhar este momento especial contigo.

Ao professor Fábio Vandresen, que me acolheu como aluno de iniciação científica no começo do curso, e compartilhou valiosos conhecimentos.

A cada um dos meus amigos, que me apoiaram nos momentos difíceis durante a graduação.

RESUMO

INSAURRALDE, D. D. R. Síntese e desenvolvimento de um simulador com realidade virtual (VR) de biodigestão para produção de biogás. 103 f. TCC (Curso de Engenharia Química), Universidade Tecnológica Federal do Paraná (UTFPR). Londrina, 2023.

O simulador imersivo que foi desenvolvido provou ser eficiente na simulação do processo de biodigestão anaeróbico de efluentes oriundos de fecularias de mandioca, fundamentado em um modelo teórico para a produção de biogás. Este modelo abrange a quantificação detalhada dos subprodutos resultantes de quinze reações bioquímicas distintas, que incluem etapas como hidrólise, acidogênese, acetogênese e metanogênese. Para enriquecer a experiência do usuário, foram integrados elementos tridimensionais criados no software Blender, tais como bombas, válvulas, o próprio biodigestor, tanques de armazenamento e trocador de calor, permitindo interações e análises em tempo real do processo. A interface do simulador é intuitiva, permitindo a manipulação através de comandos de teclado e mouse em uma perspectiva de primeira pessoa. Além disso, foi implementado um modo de realidade virtual, proporcionando uma experiência ainda mais envolvente. Os resultados obtidos indicaram uma composição de 48% de metano no biogás produzido, o que é considerado um resultado adequado para o tipo de biomassa utilizado, de acordo com a literatura. Contudo, reconhece-se a necessidade de aprimoramentos, como a inclusão de modelos termodinâmicos e cinéticos mais robustos, bem como a instalação de um sistema de aquecimento de água para melhorar a perspectiva da realidade. O simulador se destaca particularmente na simulação de tratamento de Águas Residuárias de Mandioca, no entanto, é importante notar que a precisão do simulador pode variar quando aplicado a diferentes tipos de biomassas.

Palavras-chave: Simulador. Biodigestor. Processo Industrial. Realidade Virtual.

ABSTRACT

INSAURRALDE, D. D. R. Synthesis and Development of a Virtual Reality (VR) Anaerobic Digestion Simulator for Biogas Production. 103 p. TCC (Course of Chemical Engineering) - Federal University of Technology – Paraná (UTFPR). Londrina, 2023.

The immersive simulator that was developed has proven to be effective in simulating the anaerobic digestion process of effluents from cassava starch factories, based on a theoretical model for biogas production. This model includes detailed quantification of byproducts from fifteen distinct biochemical reactions, encompassing stages such as hydrolysis, acidogenesis, acetogenesis, and methanogenesis. To enhance user experience, three-dimensional elements created in Blender software, such as pumps, valves, the biodigester itself, storage tanks and heat exchanger were integrated, allowing for real-time interactions and analyses of the process. The simulator's interface is user-friendly, enabling manipulation through keyboard and mouse commands from a first-person perspective. Furthermore, a virtual reality mode has been implemented, offering an even more immersive experience. The results indicated a biogas composition of 48% methane, which is considered an appropriate outcome for the type of biomass used, according to the literature. However, improvements are acknowledged to be necessary, including the inclusion of more robust thermodynamic and kinetic models, as well as the installation of a water heating system to enhance the perception of reality. The simulator stands out particularly in the simulation of Cassava Wastewater Treatment, however, it is important to note that the simulator's accuracy may vary when applied to several types of biomasses.

Keywords: Simulator. Anaerobic Digestion. Industrial Process. Virtual Reality.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ARFM	Água Residuária de Fecularias de Mandioca
CFD	*Computational Fluid Dynamics* (Fluidodinâmica Computacional)
CPS	Simulação de Processos Químicos
CSTR	Continuous Stirred-Tank Reactor (Reator de Tanque Agitado Contínuo)
CSV	*Comma-Separated Values* (Valores separados por vírgula)
DBO	Demanda Bioquímica de Oxigênio
DES	Simulação de Eventos Discretos
DP	Desvio Padrão
DQO	Demanda Química de Oxigênio
EDO	Equação Diferencial Ordinária
IoT	*Internet of Things* (Internet das Coisas)
Máx.	Máximo
Méd.	Médio
Mín.	Mínimo
NTK	Nitrogênio Total de Kjeldahl
PR	Estado do Paraná
SD	*System dynamics* (Dinâmica de Sistemas)
SFT	Sólidos Fixos Totais
SST	Sólidos Suspensoss Totais
SSV	Sólidos Suspensos Voláteis
ST	Sólidos Totais
SVT	Sólidos Voláteis Totais
VR	*Virtual Reality* (Realidade Virtual)
XAI	*Explainable Artificial Intelligence* (Inteligência Artificial Explicável)
XR	*Extended Reality* (Realidade Estendida)

LISTA DE SÍMBOLOS

$C_{30}H_{52}O_{26}$	Amilopectina ou Amilose
H_2O	Água
$C_6H_{12}O_6$	Glicose
$C_{13}H_{25}O_7N_3S$	Proteína
CH_4	Metano
CO_2	Dióxido de Carbono
H_3N	Amônia
H_2S	Sulfeto de Hidrogênio
$C_{57}H_{104}O_6$	Trioleína
$C_3H_8O_3$	Glicerol
$C_{18}H_{34}O_2$	Ácido Oleico
$C_2H_6O_2$	Etanol
$C_2H_4O_2$	Ácido Acético
$C_5H_7NO_2$	Microrganismo
$C_3H_6O_2$	Ácido Propanoico
$C_4H_8O_2$	Ácido Butanoico
$C_3H_6O_3$	Ácido Láctico
H_2	Hidrogênio
$\dot{n}$	Vazão molar (kmol/h)
Ca	Concentração (kg/m³)
F	Vazão de entrada de biomassa (m³/h)
MM	Massa molar (kg/kmol)
kg	Quilograma
m³	Metro cúbico
h	Hora
ξ	Extensão de reação (kmol/h)
ν	Coeficiente estequiométrico
P	Vazão de saída de biogás (kg/h)
S	Vasão de saída de digestato (kg/h)
C	Carbono
O	Oxigênio

S	Enxofre
H	Hidrogênio
N	Nitrogênio
kcal	Quilocaloria
K	Kelvin
°C	Grau celsius
CP	Capacidade térmica (kcal/kmol.K)
Q	Taxa de calor (kcal/h)
ΔT	Variação da temperatura (°C)
KJ	Quilojoule
y	Equação da reta
m	Coeficiente angular
x	Variável independente
b	Coeficiente linear
e	Exponencial
Δy	Variação de y
Δx	Variação de x
y(t)	Variável de saída da EDO
y0	Valor inicial
K	Ganho do processo
M	Variação da entrada
t	Tempo
τ	Constante de tempo

SUMÁRIO

1 INTRODUÇÃO

1.1 SIMULAÇÃO DE PROCESSOS INDUSTRIAIS NO CONTEXTO **ATUAL**

A simulação de processos é uma ferramenta poderosa e amplamente reconhecida na indústria moderna, especialmente no contexto da Indústria 4.0 e da emergente Indústria 5.0. Ela é utilizada para melhorar a eficiência da fabricação e a produtividade, além de maximizar a qualidade dos produtos. Com as mudanças nas fábricas atuais, a simulação e a modelagem de sistemas de manufatura têm se adaptado para atender a novas demandas, investigando a avaliação, avanços, práticas atuais e tendências futuras dos métodos e abordagens de simulação (Alquraish, 2022).

As ferramentas de simulação convencionais e modernas são usadas no design de sistemas de manufatura e na melhoria da produção. Desafios específicos precisam ser abordados pela comunidade de simulação, e a evolução, os avanços, as práticas atuais e as oportunidades futuras são discutidas no contexto da indústria de manufatura contemporânea. Tecnologias como gêmeos digitais, realidade virtual (VR) e Internet das Coisas (IoT) são examinadas em relação ao design de processos, planejamento e verificação (Alquraish, 2022).

DES (Simulação de Eventos Discretos) é uma técnica de modelagem e simulação que pode ser usada para analisar e projetar sistemas complexos, como manufatura, logística e sistemas de serviços. A simulação de eventos discretos é usada para modelar sistemas cujo estado muda em um momento específico, geralmente como resultado de um evento específico, como a chegada de um cliente ou uma falha de máquina (Turner; Garn, 2022).

A simulação baseada em eventos discretos (DES) e a otimização em tempo real baseada em simulação são praticadas, destacando a importância de ferramentas convencionais como programação linear, DES, Dinâmica de Sistemas (SD), Análise de Cadeia de Markov e Simulação de Monte Carlo (Alquraish, 2022). Além disso, a simulação é usada para alcançar uma produção eficaz e eficiente, melhorando processos existentes ou inovando sistemas

necessários para reduzir custos, melhorar a produtividade, alcançar competitividade e garantir qualidade.

Os desafios são encontrar ferramentas de modelagem e simulação adequadas para introduzir os conceitos do novo paradigma em cada segmento essencial da manufatura e estabelecer uma matriz de conexão entre os principais facilitadores para as Indústrias 4.0 e 5.0, particularmente pessoas, empresas e tecnologia. (Alquraish, 2022)

Turner e Garn (2022) traçam uma agenda de pesquisa para o desenvolvimento de sistemas de simulação de eventos discretos (DES), enfatizando a centralidade dos humanos nos processos de fabricação da Indústria 5.0. Eles antecipam que a integração com gêmeos digitais avançará em direção a modelos ativos e controláveis, com foco particular no enriquecimento de simulações interativas com animações interativas de realidade estendida (XR), mecanismos de jogos e inteligência artificial. O estudo aponta para a crescente importância das abordagens da tecnologia centradas no ser humano, já evidentes na Indústria 4.0, e destaca o papel da inteligência artificial explicável (XAI) na justificação de decisões tomadas por sistemas automatizados, promovendo a confiança e a transparência. É proposta uma estrutura técnica para orientar o desenvolvimento de simuladores holísticos de fabricação que integrem essas inovações e coloquem os humanos no centro dos ciclos de produção e tomada de decisão (Turner; Garn, 2022).

O software de simulação de processos químicos (CPS) é considerado uma ferramenta importante no campo da engenharia química para projetar, testar, otimizar e integrar plantas de processo. Casavant e Côté (2004) acreditam que esta tecnologia também tem grande potencial de aplicação na ecologia industrial e pode trazer contribuições significativas para o projeto e operação de ecossistemas industriais. Através do CPS, estes ecossistemas podem ser modelados, permitindo aos ecologistas industriais enfrentarem desafios técnicos e avaliar os benefícios ambientais e financeiros de diferentes cenários. Além disso, o CPS pode ser usado para resolver problemas de projeto, *retrofit* ou operacionais, identificar soluções complexas e avaliar hipóteses de maneira quantitativa (Casavant; Côté, 2004).

O CPS é particularmente valioso para modelar sistemas que ainda não

existem ou não podem ser testados na prática, abrangendo indústrias como química, petroquímica, celulose e papel, mineração, farmacêutica, produção de energia e tratamento de resíduos. Casavant e Côté (2004) ilustram o uso do CPS para criar um modelo virtual de um ecossistema industrial, que inclui múltiplas empresas e permite a avaliação de potenciais cenários de conexão entre elas. O modelo fornece informações importantes para a tomada de decisão inicial e destaca oportunidades que podem ser avaliadas posteriormente, tanto técnica como financeiramente (Casavant; Côté, 2004). Portanto, o CPS é considerado uma adição valiosa ao conjunto de ferramentas disponíveis para a ecologia industrial.

No contexto da biodigestão, a simulação é particularmente valiosa devido à complexidade dos processos biológicos e químicos envolvidos na conversão de biomassa em biogás, por exemplo. A biodigestão é um processo biológico em que matéria orgânica é decomposta por microrganismos na ausência de oxigênio (anaeróbio), resultando principalmente na produção de metano (CH_4) e dióxido de carbono (CO_2), conhecidos coletivamente como biogás. Este processo é comumente utilizado para tratar resíduos orgânicos de origens diversas, como resíduos agrícolas, lodo de estações de tratamento de esgoto, resíduos de indústrias alimentícias, entre outros.

A simulação do processo de biodigestão é importante por várias razões:

- Design e Dimensionamento: A simulação ajuda a projetar e dimensionar biodigestores para atender a demandas específicas de tratamento e produção de biogás. Isso inclui a determinação do tamanho do reator, tempo de retenção, carga orgânica, entre outros parâmetros críticos.
- Otimização: A partir da simulação, é possível otimizar as condições operacionais, como temperatura, pH, mistura e taxa de alimentação, para maximizar a produção de biogás e a estabilidade do processo.
- Controle de Processo: A simulação pode ser usada para desenvolver estratégias de controle que garantam a operação contínua e eficiente do biodigestor, mesmo diante de variações na composição do substrato ou outras perturbações externas.

- Análise Econômica: A simulação permite realizar análises econômicas, avaliando a viabilidade financeira de projetos de biodigestão, incluindo custos de capital, operacionais e potenciais receitas com a venda de biogás ou créditos de carbono.
- Impacto Ambiental: A simulação pode ajudar a entender e minimizar o impacto ambiental do processo de biodigestão, como a emissão de gases de efeito estufa e a geração de efluentes.
- Pesquisa e Desenvolvimento: A simulação é uma ferramenta essencial para a pesquisa e desenvolvimento de novas tecnologias de biodigestão, permitindo testar virtualmente inovações antes de implementá-las em escala real.
- Educação e Treinamento: A simulação pode ser usada como uma ferramenta educacional para treinar operadores e engenheiros sobre os princípios e práticas da biodigestão.

No cenário atual, a simulação de processos de biodigestão é particularmente importante devido ao crescente interesse em fontes de energia renováveis e na gestão sustentável de resíduos. A produção de biogás a partir de biodigestores não só ajuda a reduzir a dependência de combustíveis fósseis, mas também contribui para a redução de emissões de metano, um potente gás de efeito estufa, que seria liberado na atmosfera pela decomposição natural de resíduos orgânicos.

Além disso, a simulação contribui para a implementação de práticas de economia circular, onde resíduos são transformados em recursos, fechando ciclos de matéria e energia e promovendo a sustentabilidade ambiental.

1.2 SIMULADORES DE PROCESSOS IMERSIVOS

A simulação de processos industriais utilizando ferramentas 3D é uma técnica que permite a criação de ambientes virtuais interativos para treinamento e educação em operações industriais.

Os simuladores de realidade virtual em geral, são reconhecidos por oferecer uma série de benefícios e vantagens no contexto da indústria de manufatura. De acordo com autores diversos, a realidade virtual é uma

ferramenta poderosa de aprendizado, eficaz para o treinamento de trabalhadores. Ela proporciona experiências imersivas que melhoram a retenção de conhecimento e a eficácia do treinamento, permitindo aos usuários interagirem com ambientes virtuais tridimensionais. Por exemplo, os usuários podem montar modelos 3D de produtos, como um tablet modular, em um ambiente controlado e seguro (Tocu *et al.*, 2020).

Somado a isso, os simuladores imersivos oferecem uma série de benefícios e vantagens para a educação e treinamento em engenharia química e bioquímica. Os simuladores imersivos estão revolucionando a forma como os estudantes de engenharia química aprendem e se preparam para o mercado de trabalho. Essas ferramentas permitem que eles tenham uma visão realista da escala e complexidade das plantas de processo, uma oportunidade muitas vezes limitada por questões logísticas e financeiras. Em ambientes virtuais, os alunos conseguem adquirir experiência prática sem os riscos associados à operação de equipamentos em plantas reais, o que é especialmente importante devido a restrições de segurança (Kumar *et al.*, 2021, p. 143).

Além disso, os simuladores são extremamente úteis para o treinamento de operadores, tanto novatos quanto experientes, pois proporcionam um ambiente seguro para enfrentar cenários inesperados e perigosos. A sensação de estar fisicamente presente em um mundo virtual, criada por esses simuladores, pode aumentar o engajamento e ajudar na retenção de conhecimento (Kumar *et al*., 2021, p. 143).

A integração de modelos matemáticos complexos com a realidade virtual abre portas para aplicações educacionais avançadas, onde os estudantes podem interagir com simulações dinâmicas e aprender de forma mais eficaz. No entanto, para garantir que os benefícios educacionais sejam maximizados, é necessário desenvolver metodologias inovadoras para avaliar o impacto desse tipo de aprendizado. A adoção dessas tecnologias também traz implicações sociais e econômicas, exigindo investimentos em desenvolvimento e manutenção, além de uma adaptação por parte de educadores e alunos às novas tecnologias (Kumar *et al*., 2021, p. 143).

1.3 PROBLEMÁTICA

Devido a construção de um sistema de digestão anaeróbia efetivo possuir um grau elevado de complexidade, uma vez que o processo exige uma compreensão aprofundada e uma gestão assertiva de múltiplos componentes operacionais, bem como diversos desafios relacionados à segurança e questões econômicas, busca-se soluções que visa a montagem de um sistema adequado. Portanto, um simulador 3D, poderá ser capaz de modelar e otimizar o processo de biodigestão, reduzindo custos, minimizando riscos e melhorando a formação de operadores.

A seguir, destaca-se os principais desafios enfrentados na criação de um simulador 3D para biodigestores, abordando desde a complexidade do processo até as dificuldades de treinamento de pessoal especializado.

- Complexidade do processo: A biodigestão é influenciada por uma série de parâmetros operacionais, como temperatura, pH, carga orgânica, tempo de retenção e mistura, que devem ser cuidadosamente controlados para garantir a eficiência do processo.
- Custos de desenvolvimento: O projeto e a construção de biodigestores industriais exigem investimentos significativos, e erros de design podem levar a custos adicionais e atrasos na implementação.
- Riscos operacionais: A biodigestão envolve o manuseio de resíduos potencialmente perigosos e a produção de biogás, que é inflamável, exigindo medidas rigorosas de segurança.
- Dificuldade na formação e treinamento: Operadores de biodigestores precisam de treinamento especializado para gerenciar o processo de forma eficaz, o que pode ser difícil e caro de fornecer.

1.4 JUSTIFICATIVA

Com o passar do tempo, as tecnologias inovadoras se tornaram um fator importante para o progresso e eficiência de diversos setores industriais e educacionais. Com a biodigestão, um processo biológico complexo que converte

matéria orgânica em energia renovável, é importante compreender e otimizar as necessidades de cada etapa. É neste contexto que surge a lógica para a criação de um simulador 3D. Esta ferramenta tecnológica promete revolucionar a forma como projetamos, operamos e entendemos os biodigestores. Ao simular este processo num ambiente tridimensional, profissionais e estudantes podem obter informações valiosas e realizar experiências sem os custos ou riscos associados à mudança de sistemas reais. Destaca-se as principais vantagens que os simuladores 3D podem oferecer na área de biodigestão.

- Visualização e compreensão aprimoradas: Um simulador 3D permite a visualização detalhada do processo de biodigestão, facilitando a compreensão de como os diferentes componentes e parâmetros interagem.
- Testes e otimização de design: Antes da construção física, o simulador pode ser usado para testar e otimizar o design do biodigestor, identificando potenciais problemas e melhorando a eficiência operacional.
- Redução de custos e riscos: Ao permitir a simulação de cenários e a análise de riscos em um ambiente virtual, o simulador pode ajudar a evitar erros de design e operacionais que poderiam resultar em custos adicionais e riscos de segurança.
- Treinamento e educação: O simulador 3D pode ser uma ferramenta educacional poderosa, permitindo que operadores e estudantes pratiquem e aprendam sobre o processo de biodigestão em um ambiente seguro e controlado, sem os riscos associados ao treinamento em instalações reais.

1.5 OBJETIVOS

Neste projeto, busca-se como objetivo principal desenvolver e analisar um modelo interativo destinado ao processo específico da biodigestão. Para atingir esse objetivo, faremos uso do software Unreal Engine, em sua versão 5.2, desenvolvida pela Epic Games e lançada em 1998 (Unreal Engine, 2023), como

plataforma responsável pela representação visual do ambiente simulado através das tecnologias sofisticadas na área da modelagem tridimensional (3D). A concepção dessa solução tem como foco primordial disponibilizar aos alunos e profissionais envolvidos na engenharia química uma ferramenta eficaz tanto didaticamente quanto academicamente. Especificamente, visa-se:

- Através da utilização da Unreal Engine como plataforma para simulação de processos químicos, é possível aproveitar suas funcionalidades gráficas e interativas. Essas características são particularmente úteis ao criar uma representação visual dos sistemas de biodigestão.
- Criar uma representação visual precisa e educativa do processo da biodigestão, que envolve o trabalho detalhado na elaboração de modelos tridimensionais que retratem perto da realidade, os diversos elementos presentes em um biodigestor, como reatores, tubulações, tanques de armazenamento, bombas.
- Incorporar dados e Equações químicas pertinentes à biodigestão no ambiente simulado, assegurando assim que o modelo reflita comportamentos reais proporcionando uma ferramenta analítica.
- Examinar o potencial do modelo em contribuir com o design e a otimização dos biodigestores concretos, oferecendo um recurso econômico para testes experimentais bem como análise prévia das transformações procedimentais.
- Contribuir significativamente para o conhecimento atual em relação à simulação dos processos químicos ao trazer uma análise criteriosa acerca das possibilidades e limitações da utilização das tecnologias de simuladores 3D, no âmbito da engenharia química.

O objetivo deste trabalho é atingir tais metas e mostrar tanto a viabilidade técnica da utilização do Unreal Engine para simulações de engenharia química quanto fornecer perspectivas sobre como as tecnologias podem ser aplicadas visando melhorias no ensino, pesquisa e prática profissional nesse campo.

2 REVISÃO BIBLIOGRÁFICA

2.1 MATÉRIAS-PRIMAS PARA BIODIGESTÃO

Certos substratos podem apresentar dificuldades ou ser inadequados para digestão isolada devido a suas desfavoráveis relações carbono/nitrogênio ou alto teor de lipídios. Exemplos incluem resíduos de matadouros, gorduras, óleos e papel. A seleção ou combinação de substratos também deve ser considerada para otimizar outros aspectos do desempenho do biodigestor. As características físicas do digestato, especialmente sua capacidade de desaguamento, podem influenciar significativamente o balanço energético do processo. Resíduos como a polpa de beterraba açucareira, por exemplo, são notoriamente difíceis de desaguar sem o uso de produtos químicos e centrifugação, o que leva o operador a ponderar entre o custo energético do transporte de grandes volumes de digestato e o processamento do material (Banks; Heaven, 2013).

Em alguns casos, a composição química dos substratos pode resultar na precipitação de estruvita (um sal insolúvel de fosfato de magnésio e amônia) na planta de digestão, causando obstruções, danos físicos e perdas de transferência de calor. Materiais inertes densos, como areia e pedras presentes no substrato original, podem ter um efeito semelhante, e um pré-tratamento eficaz é necessário para proteger o sistema (Banks; Heaven, 2013).

Os efluentes provenientes da extração de amido de mandioca apresentam excelente potencial para digestão anaeróbica, devido à sua alta carga orgânica, representada principalmente pelos açúcares, e são gerados em grandes quantidades. Diversas pesquisas têm sido realizadas para avaliar a produção de biogás a partir da digestão desses efluentes de fecularias, variando as condições operacionais e a configuração dos reatores (Araujo *et al*., 2018)

A mandioca é uma planta originária da América do Sul e desempenha um papel crucial na alimentação em regiões tropicais, sendo uma das principais fontes de carboidratos. Durante a produção de mandioca, são gerados resíduos sólidos e líquidos que têm potencial para a produção de biogás, uma forma de bioenergia. O processamento da mandioca para extração de amido consome muita água e resulta em efluentes que são ricos em matéria orgânica e precisam

ser tratados antes de serem liberados no meio ambiente. Esses efluentes contêm substâncias cianogênicas, como a linamarina, que podem ser prejudiciais à vida aquática. O tratamento anaeróbico é uma técnica eficaz para reduzir a toxicidade desses compostos no efluente. Através da digestão anaeróbica, é possível transformar o efluente em biogás, composto principalmente por metano e dióxido de carbono, que pode ser usado para gerar calor e eletricidade para uso local. Além disso, o processo gera biofertilizante, que pode ser aplicado na agricultura, promovendo a reciclagem de nutrientes e contribuindo para a economia circular. A implementação de biodigestores representa uma solução avançada para o tratamento dos efluentes da fecularia de mandioca, alinhando-se aos princípios de sustentabilidade e fechamento de ciclos de recursos no contexto da economia circular. (Sánchez *et al*, 2017).

A Tabela 1 organiza os valores de parâmetros importantes da água residuária de fecularias de mandioca (ARFM), para trabalhar os cálculos da simulação do processo.

Tabela 1 – Variação da composição da ARFM encontrada na literatura

(continua)

PARÂMETROS	Valor Mín.	Valor Máx.	Valor Méd.	DP
PARÂMETROS GERAIS (g L^{-1})				
pH	**3,20**	**6,60**	**5,03**	**0,87**
Alcalinidade	**0,23**	**1,63**	**0,75**	**0,76**
ST	**2,83**	**63,00**	**14,00**	**16,74**
SVT	**2,25**	**52,00**	**12,43**	**14,38**
SFT	**0,20**	**11,00**	**2,73**	**4,09**
SST	**1,35**	**1,35**	**1,35**	**-**
SSV	**1,20**	**1,20**	**1,20**	**-**
DQO	**1,60**	**52,30**	**11,05**	**11,20**
DBO	**1,68**	**8,84**	**2,26**	**5,07**
DBO/DQO[a]	**0,35**	**0,55**	**0,45**	**0,14**
MACROMOLÉCULAS/ POLÍMEROS (g L^{-1})				
Carboidratos	**0,37**	**13,97**	**5,37**	**3,90**
Proteína	**2,30**	**13,50**	**7,90**	**7,92**
Gorduras	**0,63**	**5,00**	**2,82**	**3,09**
Lignina	**60,00**	**60,00**	**60,00**	**-**

Tabela 1 – Variação da composição da ARFM encontrada na literatura

(conclusão)

PARÂMETROS	Valor Mín.	Valor Máx.	Valor Méd.	DP
METABÓLITOS SOLÚVEIS (mg L^{-1})				
Maltose	**1300,00**	**1300,00**	**1300,00**	**-**
Ácido acético	**229,00**	**650,00**	**460,70**	**159,50**
Ácido lático	**158,00**	**1900,00**	**863,20**	**717,10**
Ácido propanoico	**151,00**	**151,00**	**151,00**	**-**
Ácido butírico	**1893,10**	**1893,10**	**1893,10**	**-**
Etanol	**406,00**	**589,00**	**482,50**	**88,30**
NITROGENADOS (mg L^{-1})				
NTK	**80,00**	**870,00**	**317,50**	**292,00**
Amônio (NH_4^+)	**61,50**	**61,50**	**61,50**	**-**
Nitrato (NO_3^-)	**4,00**	**4,00**	**4,00**	**-**
Cianeto livre	**3,50**	**43,70**	**30,60**	**16,60**
Cianeto total	**444,00**	**800,00**	**622,00**	**251,70**
NUTRIENTES (mg L^{-1})				
Potássio	**340,00**	**1863,50**	**645,50**	**504,20**
Magnésio	**24,00**	**700,00**	**237,40**	**237,50**
Cálcio	**6,90**	**3100,00**	**518,20**	**950,90**
Fósforo	**20,00**	**620,00**	**194,90**	**203,50**
Ferro	**2,50**	**24,00**	**13,20**	**7,50**
Sódio	**0,50**	**485,00**	**132,90**	**207,20**
Cloro	**52,00**	**52,00**	**52,00**	**-**
Bromo	**30,00**	**30,00**	**30,00**	**-**
Cobre	**0,20**	**1,10**	**0,60**	**0,70**
Zinco	**2,00**	**4,20**	**3,10**	**1,60**
Manganês	**3,70**	**3,70**	**3,70**	**-**
Enxofre	**19,50**	**19,50**	**19,50**	**-**
Boro	**5,00**	**5,00**	**5,00**	**-**
OUTROS ÍONS (mg L^{-1})				
Difosfato (HPO_4^{2-})	**47,30**	**47,30**	**47,30**	**-**
Sulfato (SO_4^{2-})	**63,8**	**63,8**	**63,8**	**-**

Fonte: TAIATELE JUNIOR, 2023

2.2 PROCESSOS DE TRANSFORMAÇÃO NA BIODIGESTÃO

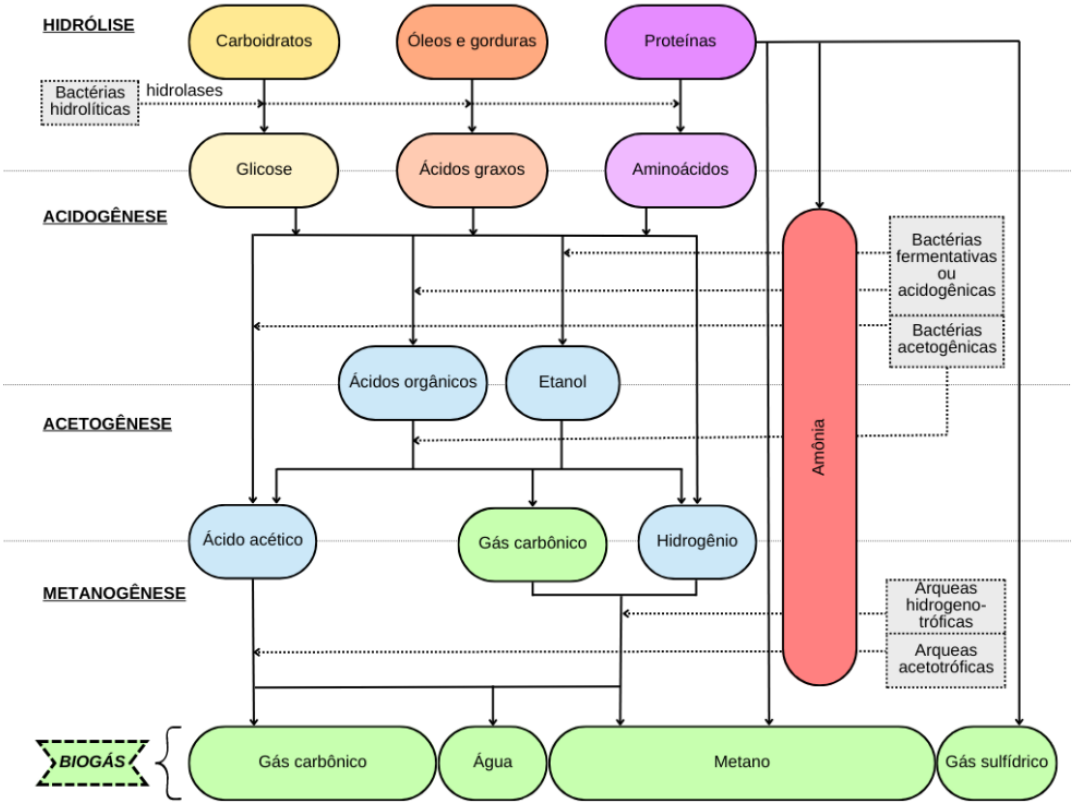
A digestão anaeróbica é um processo amplamente reconhecido como uma alternativa interessante para o tratamento e estabilização de substratos orgânicos residuais. No entanto, várias limitações técnicas foram observadas com base nas características da matéria orgânica submetida ao processo, como a presença de altas concentrações de açúcares solúveis ou gorduras. A tecnologia de digestão anaeróbica em múltiplas etapas é descrita como uma opção viável no controle de variáveis, otimizando as condições ambientais dos principais microrganismos envolvidos no processo, garantindo alta remoção de sólidos e produção de metano, além de permitir um maior rendimento energético através da geração de hidrogênio molecular como combustível. Vários estudos revisaram o processo de digestão anaeróbica em múltiplas etapas no tratamento de resíduos alimentares, embora poucos relatem seu uso aplicado diretamente a resíduos agroindustriais (Cremonez *et al.*, 2021).

O processo de digestão anaeróbica é realizado por um grupo de microrganismos e geralmente dividido em quatro fases: hidrólise, acidogênese, acetogênese e metanogênese. A hidrólise, ou a primeira etapa do processo, é caracterizada pela decomposição de compostos complexos e de alto peso molecular (proteínas, gorduras e carboidratos) em moléculas solúveis, como aminoácidos, ácidos graxos e açúcares de cadeia curta. Esta ocorre através da ação de enzimas extracelulares liberadas pelas bactérias presentes no meio (Cremonez *et al.*, 2021).

Na etapa de acidogênese, os substratos de baixo peso molecular obtidos da fase de hidrólise são absorvidos pelas bactérias e utilizados no ciclo metabólico desses microrganismos, com a excreção de ácidos orgânicos voláteis (ácidos acético, propiônico, lático e fórmico, entre outros), bem como alguns álcoois e gases como dióxido de carbono e hidrogênio molecular. Na acetogênese, os ácidos orgânicos gerados a partir da fase de acidogênese são reduzidos a ácido acético. Este processo é realizado por um grupo restrito de microrganismos homoacetogênicos. Na fase final, a metanogênese, dois grupos de microrganismos metanogênicos são responsáveis pela conversão dos compostos intermediários, até então produzidos, em biogás. Os metanogênicos acetotróficos produzem metano pelo uso de ácido acético, enquanto os

hidrogenotróficos geram metano pelo uso de dióxido de carbono e hidrogênio (Cremonez *et al.*, 2021).

Figura 1 – Etapas e respectivas rotas metabólicas da digestão anaeróbia.



Fonte: TAIATELE JUNIOR, 2023

A eficiência do processo está diretamente relacionada à colaboração sinérgica dos microrganismos presentes em todas as etapas. No processo de digestão, a hidrólise desempenha um papel determinante ao limitar a disponibilidade de moléculas solúveis para as células bacterianas. Porém, caso o substrato usado seja constituído por compostos facilmente fermentáveis, a etapa metanogênica pode ser a fase que limite o processo uma vez que os microrganismos destes são bastante sensíveis ao acúmulo de ácidos no meio (Cremonez *et al.*, 2021).

Antes de especificar as reações químicas que envolvem o processo de biodigestão, é importante destacar informações pertinentes ao amido, um carboidrato chave presente nas águas residuais da indústria de processamento de mandioca, sendo crucial para o processo mencionado. Este polissacarídeo é constituído por dois tipos de moléculas: amilopectina, que possui uma estrutura

ramificada devido às ligações glicosídicas α-1,4 e α-1,6, e amilose, que é linear e contém apenas ligações α-1,4. A estrutura molecular desses componentes influencia a eficiência da biodigestão, pois afeta a velocidade com que podem ser quebrados por microrganismos (Taiatele, 2023).

Nas simulações computacionais, muitos compostos não têm representações diretas, o que requer adaptações. Em seu trabalho, Taiatele (2023) utilizou a amilopectina, com uma massa molar de 828,727 kg.kmol^{-1} e fórmula molecular $C_{30}H_{52}O_{26}$. A mesma configuração foi replicada para a amilose, sendo configurada no software Aspen Plus com um componente análogo chamado maltopentaose (Taiatele, 2023).

Conforme Taiatele (2023), as reações químicas de cada etapa da biodigestão para o substrato em questão podem ser representadas por Equações numeradas de 1 a 15. Essas Equações estão ordenadas de acordo com o tipo de reação, embora não ocorram necessariamente na ordem descrita, a literatura simplifica as informações para proporcionar uma melhor ilustração, no entanto, é importante ressaltar que a realidade pode diferir significativamente. As reações serão implementadas no simulador e desempenharão um papel crucial nos cálculos preditivos para a geração de biogás.

Reações da etapa de hidrólise:

$$C_{30}H_{52}O_{26} + 4H_2O \rightarrow 5C_6H_{12}O_6 \quad (1)$$

$$C_{30}H_{52}O_{26} + 4H_2O \rightarrow 5C_6H_{12}O_6 \quad (2)$$

$$C_{13}H_{25}O_7N_3S_1 + 6H_2O \rightarrow 6{,}5CH_4 + 6{,}5CO_2 + 3H_3N + H_2S \quad (3)$$

$$C_{57}H_{104}O_6 + 3H_2O \rightarrow C_3H_8O_3 + 3C_{18}H_{34}O_2 \quad (4)$$

$$C_6H_{12}O_6 \rightarrow 2C_2H_6O + 2CO_2 \quad (5)$$

$$2C_2H_6O + CO_2 \rightarrow 2C_2H_4O_2 + CH_4 \quad (6)$$

Reações da etapa de acidogênese:

$$C_6H_{12}O_6 + 0{,}1115H_3N \rightarrow 0{,}1115C_5H_7O_2N_1 + 0{,}7440C_2H_4O_2 + 0{,}5C_3H_6O_2 + 0{,}4409C_4H_8O_2 + 0{,}6909CO_2 + 1{,}0254H_2O \quad (7)$$

$$C_6H_{12}O_6 \rightarrow 2C_3H_6O_3 \quad (8)$$

$$3C_3H_6O_3 \rightarrow 2C_3H_6O_2 + C_2H_4O_2 + CO_2 + H_2O \quad (9)$$

$$C_3H_8O_3 + 0{,}0407H_3N + 0{,}0291CO_2 + 0{,}0005H_2 \rightarrow 0{,}0407C_5H_7O_2N_1 + 0{,}9418C_3H_6O_2 + 1{,}0931H_2O \quad (10)$$

Reações da etapa acetogênese:

$$C_{18}H_{34}O_2 + 0{,}1701H_3N + 0{,}2501CO_2 + 15{,}2396H_2O \rightarrow 0{,}1701C_5H_7O_2N_1 + 8{,}6998C_2H_4O_2 + 14{,}4978H_2 \quad (11)$$

$$C_4H_8O_2 + 0{,}0653H_3N + 0{,}8038H_2O + 0{,}0006H_2 + 0{,}5543CO_2 \rightarrow 0{,}0653C_5H_7O_2N_1 + 1{,}8909C_2H_4O_2 + 0{,}4460CH_4 \quad (12)$$

$$C_3H_6O_2 + 0{,}0620H_3N + 0{,}3143H_2O \rightarrow 0{,}0620C_5H_7O_2N_1 + 0{,}9345C_2H_4O_2 + 0{,}6604CH_4 + 0{,}1607CO_2 + 0{,}0006H_2 \quad (13)$$

Reações da etapa metanogênese

$$C_2H_4O_2 + 0{,}0220H_3N \rightarrow 0{,}0220C_5H_7O_2N_1 + 0{,}9450CH_4 + 0{,}9450CO_2 + 0{,}0660H_2O \quad (14)$$

$$14{,}4976H_2 + 3{,}8334CO_2 + 0{,}0836H_3N \rightarrow 0{,}0836C_5H_7O_2N_1 + 3{,}4154CH_4 + 7{,}4996H_2O \quad (15)$$

2.3 MODELAGEM E SIMULAÇÃO

Serpa *et al.* (2020) descreve o desenvolvimento de um jogo baseado em simulação para o treinamento de operadores de equipamentos pesados na indústria de manufatura de placas de aço. O jogo foi desenvolvido em colaboração com uma empresa do setor de eletrodomésticos e utiliza a Unreal Engine 4 e o framework Playfab para criar um ambiente de aprendizado que simula o processo de estampagem de chapas de aço. O objetivo é fornecer uma compreensão global do processo de produção para todos os funcionários, em vez de se concentrar apenas em trabalhadores de nível básico ou gerentes. A simulação permite que os operadores testem diferentes métodos e cronogramas de operação dos equipamentos de forma segura e econômica, sem interromper a linha de produção real. Além disso, o jogo utiliza dispositivos de controle físicos semelhantes aos usados na fábrica para melhorar a habilidade técnica e o conhecimento procedimental dos operadores (Serpa *et al.*, 2020).

Uma avaliação de experiência do usuário foi realizada com 13 operadores de equipamentos pesados em seu ambiente industrial real. Os resultados mostraram que métodos baseados em simulação podem ser usados para criar um ambiente de trabalho motivacional e tornar o aprendizado mais divertido e eficaz (Serpa *et al.*, 2020).

Solmaz *et al*. (2023) desenvolveram um ambiente de aprendizagem imersivo em realidade virtual denominado "Virtual Garage", que incorpora simulações de dinâmica de fluidos computacional (CFD) para enfrentar desafios na educação em engenharia. O estudo envolveu 24 estudantes de pós-graduação que avaliaram a usabilidade, experiência do usuário, carga de tarefas e problemas do simulador por meio de questionários padronizados, métricas auto-relatadas e uma entrevista semiestruturada. Os resultados indicaram que o "Virtual Garage" foi bem recebido pelos participantes, e os autores identificaram características que poderiam melhorar a qualidade da experiência em VR com simulações de CFD. As implicações foram incorporadas ao estudo para fornecer orientações práticas para desenvolvedores e profissionais da área (Solmaz *et al*. 2023).

O estudo revelou que a experiência imersiva em realidade virtual pode reduzir barreiras de entrada para assuntos de aprendizagem cognitivamente complexos e pode desencadear habilidades cognitivas avançadas com interações espaciais e conteúdo técnico de fácil acesso. O "Virtual Garage" é uma experiência imersiva holística que educa os alunos por meio de um problema de engenharia da vida real resolvido com dados de simulação de CFD. O protótipo testado mostrou que o ambiente de aprendizagem imersivo pode ser uma ferramenta valiosa para a educação em engenharia, oferecendo uma experiência de usuário envolvente e interativa (Solmaz *et al*. 2023).

Os autores destacam a importância do design instrucional adequado e da consideração da carga cognitiva para evitar sobrecarga dos alunos ao interagir com dados de simulação. Eles sugerem que ambientes de aprendizagem com simulações de CFD devem ser adequadamente estruturados, levando em conta componentes relevantes do design instrucional. Além disso, o estudo sugere que a visualização de dados de CFD em VR pode proporcionar uma interface melhor do que as telas 2D para trabalhar com

conjuntos de dados complexos no contexto da visualização científica. O estudo também aponta para a necessidade de mais pesquisas sobre avaliações de protótipos, incluindo fatores humanos, para fornecer melhores aplicações e desbloquear o potencial dessas ferramentas para estudantes e formuladores de políticas. A pesquisa destaca a importância de avaliar fatores humanos em experiências educacionais imersivas em VR com simulações de CFD antes de se concentrar nos resultados de aprendizagem (Solmaz *et al*. 2023).

Outro estudo realizado por Silva *et al* (2022) avalia o benefício de tecnologia imersiva no processo educacional, analisando ondas cerebrais em ambiente de realidade virtual para fornecer dados que ajudem a otimizar novas tecnologias para este contexto. Observou-se aumento notável nos ritmos Beta no eletroencefalograma (EEG), ao usar o dispositivo VR, que são as ondas cerebrais associadas ao foco. Contudo, para concluir sobre a avaliação da taxa de aprendizado e retenção de conhecimentos pelos voluntários, o estudo carece de teste (Silva *et al*, 2022).

Além disso, é destacado que a gamificação pode melhorar significativamente o desempenho de aprendizagem e aumentar a atividade cerebral em áreas relacionadas à atenção e à memória de trabalho. Isso sugere que ambientes altamente imersivos que utilizam gamificação, como a RV, podem se beneficiar desse aumento de atenção e memória de trabalho, características importantes para o aprendizado. Portanto, o experimento apresentado reforça a noção de que VR para fins educacionais deve ser interativa e imersiva para ser eficaz (Silva *et al*, 2022).

Baseado nos estudos apontados, compreende-se que a justificativa para desenvolver simuladores imersivos, em diversos contexto, é sustentada por uma variedade de objetivos. Essa abordagem tem o potencial de proporcionar resultados expressivos em relação à experiência para a qual o simulador é proposto. Para o contexto deste trabalho, gerar uma experiência imersiva na produção de biogás, pode aprimorar os conhecimentos dos participantes sobre o processo em si, gerar *insights* de resultados sobre a montagem dos equipamentos, aproximar o público em geral para próximo de processos industriais, através de uma experiência enriquecedora. Adicionado a isso, tem-se o aumento na eficiência de treinamentos, permitindo que os participantes

adquiram habilidades práticas mais rápidas e reduzindo custos.

3 METODOLOGIA

O processo de desenvolvimento do simulador de biodigestão proposto neste trabalho será estruturado em três fases interdependentes, cada uma desempenhando um papel crucial na construção do simulador. Inicialmente, a etapa de Modelagem 3D será o momento para criação de elementos 3D envolvidos no processo de biodigestão, visando uma visualização realista e detalhada. Na sequência, a resolução de cálculos realizada programaticamente, resolvendo Equações e criando algoritmos necessários para simular de maneira precisa a produção de biogás, garantindo precisão nos resultados obtidos. Por fim, a fase de movimentação e visualização abordará a implementação de mecanismos que permitam a interatividade do usuário com o simulador, bem como a apresentação visual intuitiva e informativa dos dados gerados durante a simulação, promovendo uma experiência imersiva.

As ferramentas tecnológicas para atingir o resultado esperado:

- Blender – Modelagem 3D
- Unreal Engine 5 – Movimentação e Interação
- Blueprint – Programação de interação 3D.
- Linguagem de Programação C++ - Cálculos e algoritmos

3.1 MODELAGEM DE EQUIPAMENTOS 3D

A modelagem 3D será realizada pelo software Blender, o qual é responsável pela criação de componentes 3D gratuito e de código aberto que oferece um conjunto robusto de ferramentas para modelagem, animação, renderização, pós-produção, criação e edição de vídeo, e simulação de física. Desenvolvido por Ton Roosendaal, co-fundador da empresa NeoGeo na época, o Blender teve sua origem como a reescrita de um outro software 3D utilizado pela própria empresa. Iniciado como um projeto mais modesto em 1995, o Blender cresceu para se tornar uma das ferramentas de modelagem e animação 3D mais populares disponíveis hoje (Blender, 2023). Seu principal público são

artistas independentes, estúdios de pequeno e médio porte, e educadores devido à sua licença livre e à sua capacidade de competir com softwares comerciais caros. Blender é frequentemente elogiado por sua comunidade ativa, que contribui com uma variedade de plugins e extensões que expandem ainda mais suas capacidades.

Além disso, muitos profissionais usam o Blender para criar ativos 3D que são posteriormente importados e utilizados dentro do Unreal Engine, demonstrando como essas ferramentas podem ser complementares no desenvolvimento de produtos 3D.

Os equipamentos que serão modelados em 3D, serão os seguintes:

- Tanque de armazenamento de água quente
- Tanque de armazenamento de substrato
- Bombas e Compressor
- Trocador de calor
- Biodigestor
- Agitador
- Válvulas
- Tubulações
- Casa de gás e cilindros

Para modelagem 3D, as dimensões não são relevantes, uma vez que o objetivo em si é causar a impressão de que os equipamentos possuem um tamanho razoável em relação ao personagem de simulação, além do software permitir aumentar a escala sempre que possível. Portanto esta modelagem tem um efeito qualitativo, ao invés de quantitativo.

Além disso, o software exige um grau elevado de conhecimento para modelagem 3D, portanto alguns modelos serão exportados de opções gratuitas e de livre uso quando for conveniente, sem causar nenhum dano a propriedade do simulador de modo geral.

O modelo do tanque para armazenamento do substrato e da água será tubular horizontal e metálico, conforme Figura 2, modelo ideal para armazenar líquidos. A escolha visa pela praticidade na modelagem 3D, portanto o mesmo

modelo será utilizado para ambos os fluidos, mas devidamente identificados.

Figura 2 – Exemplo de tanque horizontal

Fonte: TANQUESEBOMBAS, 2023

Para transportar os líquidos armazenados nos tanques, pela tubulação até o trocador de calor serão utilizadas bombas centrífugas, conforme Figura 3, devido ao seu modelo característico para visualização 3D, além de amplamente ser aplicado em diversos contextos industriais, e sua característica de compatibilidade com diversos fluidos.

Figura 3 – Exemplo de bomba centrífuga

Fonte: EBOMBAS, 2023

Para realizar a simulação 3D do transporte e armazenamento do biogás, optou-se pela implementação de um compressor, conforme Figura 4. Essa decisão visa, sobretudo, proporcionar uma representação mais realista e dinâmica no ambiente virtual, permitindo a visualização detalhada do processo

de compressão do biogás antes de seu armazenamento em cilindros metálicos.

Figura 4 – Exemplo de compressor

Fonte: INDUSTRIALIS, 2023

Com o objetivo de preparar o substrato antes de iniciar a biodigestão, mantendo-o a uma temperatura em 55ºC, é necessário a inserção de um trocador de calor casco-tubo, conforme Figura 5, na configuração contracorrente, de maneira que a troca térmica seja mais eficiente, uma vez que nesta configuração há um máximo aproveitamento da diferença de temperatura ao longo do trocador de calor.

Figura 5 – Exemplo de trocador de calor casco tubo

Fonte: TERMOTEK, 2023

Para a biodigestão, foi selecionado o biodigestor no modelo de reator de

tanque agitado contínuo (CSTR), uma vez que este modelo para fins de simulação em 3D, possui um resultado melhor, em questão de visualização do processo, além de ser amplamente utilizado. Um dos modelos CSTR que há no Brasil atualmente, está localizado em Tamboara, PR, na empresa Geo Biogás conforme Figura 6.

Figura 6 – Biodigestor CSTR Geo Elétrica Tamboara

Fonte: AMANHA, 2023

Para realizar a agitação no interior do biodigestor, o agitador escolhido de acordo com a Figura 7, o qual segundo o fornecedor (Ovivowater, 2023), é ideal para este tipo de processo. Este processo não será simulado, tendo apenas uma representação visual.

Figura 7 – Exemplo de agitador para biodigestão

Fonte: OVIVOWATER, 2023

A válvula para controle de vazão escolhida é a válvula guilhotina, conforme Figura 8, o qual, para fins de simulação 3D, possui uma representação melhor e fácil visualização para ponto de contato.

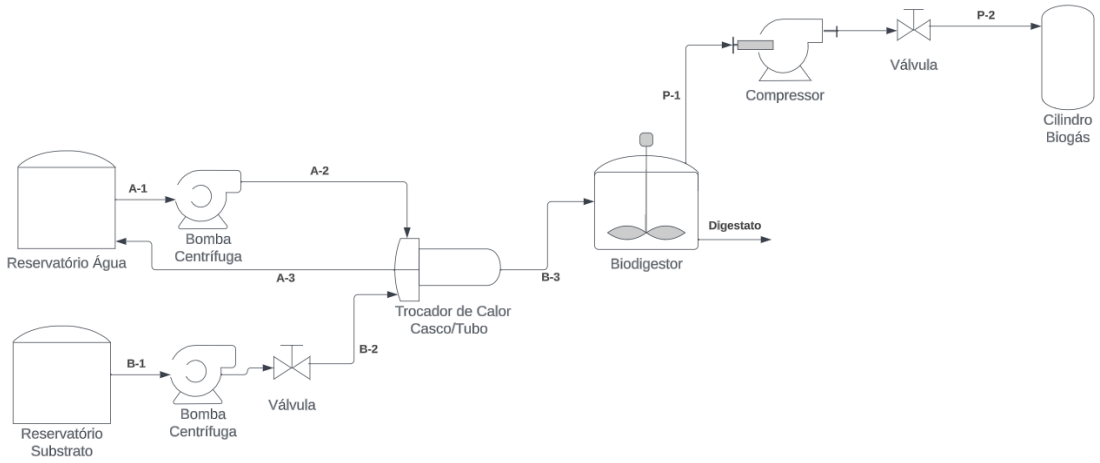
Figura 8 – Exemplo de válvula guilhotina

Fonte: VALMEC, 2023

As tubulações serão representadas de forma genérica e com efeito metálico, sem muito detalhe aparente. A casa de gás e os cilindros, têm efeito somente visual, portanto, não serão utilizados para fins de cálculos de simulação.

Após a modelagem 3D, os equipamentos serão organizados conforme Figura 9, de forma que o biogás bruto saia pelo topo do biodigestor, e o digestato pelo fundo.

Figura 9 – Layout dos equipamentos para simulação de biodigestão



Fonte: Autor, 2023

3.2 MODELO DE SIMULAÇÃO PARA GERAÇÃO DE BIOGÁS

3.2.1 Composição da matéria-prima

Inicialmente, será elaborada uma tabela de dados referente aos componentes presentes no resíduo escolhido, no caso, água residuária de fecularias de mandioca. Essa tabela servirá como fonte de alimentação para o biodigestor, dando início ao processo de digestão anaeróbica. De acordo com os dados fornecidos por Taiatele em 2023, a composição será organizada conforme os parâmetros apresentados na Tabela 2.

Tabela 2 – Composição de ARFM em kg/m³

Nome	Fórmula	Concentração na entrada (kg/m³)	Massa Molar
Amilopectina	$C_{30}H_{52}O_{26}$	4,292	828,727
Amilose	$C_{30}H_{52}O_{26}$	1,073	828,727
Água	H_2O	980,008	18,0153
Etanol	$C_2H_6O_2$	0,483	46,069
Ácido Acético	$C_2H_4O_2$	0,461	60,0526
Proteína	$C_{13}H_{25}O_7N_3S$	7,900	367,42
Amônia	H_3N	0,062	17,0306
Trioleína	$C_{57}H_{104}O_6$	2,815	885,449
Ácido Propanoico	$C_3H_6O_2$	0,151	74,0794
Ácido Butanoico	$C_4H_8O_2$	1,893	88,1063
Ácido Láctico	$C_3H_6O_3$	0,863	90,0788

Fonte: Elaborado pelo autor com base em Taiatele, 2023.

Os dados sobre a proteína adicionada ($C_{13}H_{25}O_7N_3S$) representam, de maneira básica, o conjunto de proteínas solubilizadas em um efluente industrial. Essas informações foram obtidas de uma simulação feita por Rajendran et al. (2014). Segundo Taiatele (2023), os dados referentes a Amilose e Amilopectina, podem ser descritos conforme estão na Tabela 2, utilizando a fórmula molecular do composto maltopentaose.

Além dos componentes de entrada, pode-se descrever as propriedades utilizadas de todos as substâncias geradas pelo processo de biodigestão, conforme Tabela 3.

Tabela 3 – Compostos químicos presentes no processo de biodigestão

Nome	Fórmula	Massa molar
Glicose	$C_6H_{12}O_6$	180,158
Dióxido de Carbono	CO_2	44,010
Metano	CH_4	16,043
Sulfeto de Hidrogênio	H_2S	34,082
Glicerol	$C_3H_8O_3$	92,095
Ácido Oleico	$C_{18}H_{34}O_2$	282,467
Microrganismo	$C_5H_7NO_2$	113,116
Hidrogênio	H_2	2,016

Fonte: Elaborado pelo autor com base em Taiatele, 2023.

Além disso, é necessário a adição de microrganismos na simulação, para representação deste é representado pela fórmula genérica $C_5H_7NO_2$, conforme Rajendran et al. (2014) descreve.

Os dados destes compostos serão importados para Unreal Engine, em arquivo de formato CSV e posteriormente lidos no algoritmo de cálculo da simulação que será descrito nas próximas sessões.

Além desses valores, tem-se os dados de fração líquida e gasosa para cada componente, fornecido pelo software Aspen Plus e descrito por Taiatele (2023), conforme Tabela 4, de forma que quantifique a quantidade de matéria de cada composto, presentes na fase líquida e gasosa. A fração líquida, embora se refiram a componente gasosos, representam que parte desses compostos estão sendo quebrados em outras moléculas não conhecidas formando a composição do digestato.

Tabela 4 – Tabela de fração líquida e gasosa de compostos

Composto	**Fração Líquida**	**Fração Gasosa**
Água	0,9978	0,0022
Dióxido de Carbono	0,0247	0,9753
Metano	0,0013	0,9987
Amônia	0,9615	0,0393
Sulfeto de Hidrogênio	0,0757	0,9243

Fonte: Elaborado pelo autor com base em Taiatele, 2023.

3.2.2 Modelo de geração de biogás

O modelo proposto por Taiatele (2023) tem como fundamento um reator estequiométrico, sendo empregado como base para realizar os cálculos de simulação. Esses cálculos abrangem cada reação química envolvida. As configurações específicas de cada reação química, detalhadas na seção 2.2, estão disponíveis na Tabela 5.

Tabela 5 – Dados de conversão para cada reação descrita na seção 2.2

Tipo de reação	Número da Reação	Fator de conversão
HIDRÓLISE	1	1,00
HIDRÓLISE	2	1,00
HIDRÓLISE	3	0,54
HIDRÓLISE	4	1,00
HIDRÓLISE	5	1,00
HIDRÓLISE	6	1,00
ACIDOGÊNESE	7	0,33
ACIDOGÊNESE	8	0,50
ACIDOGÊNESE	9	1,00
ACIDOGÊNESE	10	1,00
ACETOGÊNESE	11	1,00
ACETOGÊNESE	12	1,00
ACETOGÊNESE	13	1,00
METANOGÊNESE ACETOTRÓFICA	14	1,00
METANOGÊNESE HIDROGENOTRÓFICA	15	1,00

Fonte: Elaborado pelo autor com base em Taiatele, 2023.

Cada etapa da reação, prevê o cálculo da vazão molar baseado na concentração de cada componente na corrente de entrada e a vazão de entrada em $m^3.h^{-1}$, cálculo da extensão de reação baseado no reagente limitante e a vazão molar de saída, pelas Equações 16, 17 e 18, respectivamente.

$$\dot{n} = Ca.\frac{F}{MM} \tag{16}$$

Onde $\dot{n}$ é a vazão molar em $kmol.h^{-1}$, Ca é a concentração de cada composto na corrente de entrada em $kg.m^{-3}$, F é a vazão de entrada em $m^3.h^{-1}$ e MM é a massa molar do componente em $kg.kmol^{-1}$.

$$\xi = \frac{\dot{n}_{limitante}}{\upsilon_{limitante}} \quad (17)$$

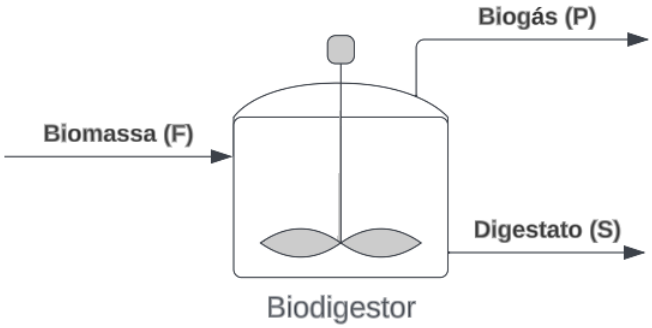
Onde ξ é a extensão de reação em kmol.h^{-1}, $\dot{n}$ é a vazão molar do reagente limitante em kmol.h^{-1}, e υ é o coeficiente estequiométrico do reagente limitante.

$$\dot{n}_f = \dot{n}_\iota + \xi.\upsilon \quad (18)$$

Onde $\dot{n}_f$ é a vazão molar de saída em kmol.h^{-1}, $\dot{n}_\iota$ é a vazão molar de entrada em kmol.h^{-1}, calculado para cada composto.

O processo de biodigestão é composto pela entrada de biomassa (F) e a saída dos produtos, sendo biogás (P) pelo topo e digestato (S) pelo fundo, conforme Figura 10.

Figura 10 – Visualização do processo de biodigestão



Fonte: Autor, 2023.

Dessa forma, tem-se o balanço de massa global pela Equação 19, em kg.h^{-1}.

$$F = P + S \quad (19)$$

Onde F é a entrada de biomassa, o qual, foi escolhido a água residuária de fecularias de mandioca e a sua composição está descrita na Tabela 2. P é a saída de biogás, e S se refere a saída de digestato, onde os compostos majoritários previstos em sua composição estão descritos na Tabela 6.

Ao final, obtém-se a composição, para o digestato e para o biogás,

conforme descrito pela Tabela 6, em que para o biogás contém compostos gasosos definidos, e para o digestato, tem-se a presença de elementos, que formam outros compostos que são desconhecidos pela literatura, portanto são apontados em sua fase elementar para ilustrar a quantidade posteriormente.

Tabela 6 – Compostos e elementos majoritários presente no produto e subproduto

Digestato	Biogás
Água	Água
Proteína	CO_2
C	CH_4
O	NH_3
S	H_2S
H	-
N	-

Fonte: Elaborado pelo autor com base em Taiatele, 2023.

Vale destacar, que Taiatele (2023), validou a simulação feita pelo Aspen Plus e um modelo escrito em Excel. O modelo do Aspen Plus prevê uma robustez maior devido a utilização de outras propriedades físico-químicas, e reações que envolvem formação de sais. No entanto, este trabalho tratará somente dos cálculos envolvidos no modelo em Excel, e estas outras propriedades não serão incorporadas ao simulador.

3.2.3 Modelo de balanço de energia para trocador de calor

Optou-se pelo trocador de calor casco-tubo para o processo, fundamentando-se no balanço de energia e massa. O objetivo é determinar a quantidade adequada de água para aquecer a biomassa à temperatura ideal para o procedimento, estabelecida em 55 ºC, conforme indicado por Taiatele (2023). O autor também fornece os dados de capacidade térmica (CP) para a biomassa, calculados pelo Aspen Plus, com um valor de 18,048 $kcal.kmol^{-1}.K^{-1}$.

Para encontrar o CP da água nessa unidade, obteve-se o valor de 1,0044 $cal.kg^{-1}K^{-1}$ a 90°C (The Engineering ToolBox, 2004) e multiplicou pelo valor de sua massa molar (Tabela 2) e obteve-se o valor de 18,094 $kcal.kmol^{-1}.K^{-1}$, a temperatura de 90ºC. Ambos os CPs serão considerados constantes, visto que a variação de temperatura é suficientemente baixa para desconsiderar a alteração do CP. Isso ocorre porque a biomassa possui propriedades

semelhantes às da água, devido ao elevado teor de água em sua composição.

Para os cálculos, será utilizada a temperatura de entrada da biomassa a 25ºC (temperatura ambiente). Além disso, será considerado que a água entra aquecida a 90ºC e se resfria para atingir 65ºC. Dessa forma, é possível encontrar o calor envolvido no processo pela Equação 20.

$$\dot{Q} = \dot{n}.cp.\Delta T \tag{20}$$

Onde $\dot{Q}$ é a taxa de calor do processo em kcal.h^{-1}, $\dot{n}$ é a vazão molar em kmol.h^{-1}, CP é a capacidade térmica do composto em kcal.kmol^{-1}.K^{-1} e ΔT a variação de temperatura (saída-entrada) em ºC, uma vez que a variação de temperatura não afetará os resultados por conta da unidade utilizado, não sendo preciso inserir na unidade Kelvin. O calor posteriormente será convertido para KJ/h, multiplicando-o por 4,184.

E então, encontrar o valor de mols necessária de água pela Equação 21.

$$\dot{n} = \frac{\dot{Q}}{cp.\Delta T} \tag{21}$$

A implementação dos cálculos do trocador de calor em código, podem ser acessados através do Apêndice F.

3.2.4 Transformação do modelo em algoritmos de programação

O modelo usado para calcular o reator estequiométrico será traduzido para a linguagem de programação que o autor domina, visando entender e desenvolver algoritmos para cada etapa do processo. Dada a alta complexidade, utilizar a linguagem de programação nativa da Unreal Engine pode tornar o processo longo e demorado. A estratégia adotada é criar o algoritmo inicial em Python e, posteriormente, convertê-lo para C++, que é compatível com a Unreal Engine. No entanto, a Unreal dispõe duas formas de trabalhar com programação, a Blueprint e C++, discutidos no próximo tópico.

3.2.5 Blueprint e C++

A Unreal Engine fornece recursos programáticos para criação de interação, movimentação, animação e tudo o que é preciso para um simulador imersivo, sem a necessidade de recorrer a recursos externos, com exceção da modelagem 3D. Esses recursos programáticos são linguagem de programação C++ e programação por script Blueprint, ambas permitem a implementação da lógica de programação de aspectos visuais e interatividade, no entanto, cada uma com sua particularidade (Zaytsev, 2023).

C++ é uma linguagem de programação, que permite o gerenciamento detalhado sobre a memória, essencial para o desenvolvimento de sistemas e aplicações que demandam alto desempenho. Caracterizada por sua eficiência e flexibilidade, C++ suporta paradigmas de programação como procedural, orientado a objetos e genérico, tornando-a uma escolha robusta para a criação de software complexo (Wikipédia, 2023). Além disso, são boas práticas na linguagem C++, construir códigos de programação em dois tipos de arquivos, na extensão “.h”, que é referente a cabeçalho, e “.cpp”, onde, no primeiro é configurado o protótipo de todas os métodos de classes, sem muito detalhes, e no segundo, é realizado a implementação de fato, contendo todos os detalhes do código.

No contexto da Unreal Engine, é oferecido uma estrutura rica escrita em C++, permitindo aos desenvolvedores estenderem e personalizar a funcionalidade do motor de jogo para atender às suas necessidades específicas. A integração de C++ na Unreal Engine possibilita a criação de mecânicas de simulador imersivo avançadas, interações complexas e simulações realistas, aproveitando a velocidade e o controle que C++ oferece. Já o Blueprint, desenvolvido pela própria empresa, é um ambiente visual de *scripting*, que trabalha em conjunto com o código C++, permitindo aos desenvolvedores prototipar rapidamente e iterar em suas ideias, mesmo sem um conhecimento profundo de C++ (Zaytsev, 2023). A combinação de C++ e Unreal Engine é, portanto, uma poderosa aliança para o desenvolvimento de simuladores imersivos e aplicações interativas de alta qualidade.

Ambas apresentam vantagens e desvantagens. A linguagem C++ é mais complexa, demandando conhecimentos profundos de programação e

computação em geral, o que representa um desafio para aqueles com baixo domínio nessa área (Zaytsev, 2023). Todavia, proporciona elevado desempenho, acelerando e otimizando os algoritmos, fator crucial no contexto dos simuladores imersivos. Em contraste, o Blueprint possui uma curva de aprendizado relativamente menor, uma aceitação mais ampla e simplifica a criação intuitiva de algoritmos (Zaytsev, 2023). Contudo, isso implica em um custo, uma vez que o consumo de recursos computacionais, como memória e tempo de processamento, é mais elevado. Sem devidos cuidados, isso pode inviabilizar um software devido a este alto custo. Existe a possibilidade de incorporar conteúdo desenvolvidos em C++ no Blueprint, mas o contrário não é verdadeiro. No contexto deste trabalho, não há uma grande preocupação nesse aspecto, pois o projeto é considerado mais modesto em recursos 3D e apresenta pouca interatividade.

Os algoritmos para os cálculos de simulação do biodigestor serão implementados em C++. Apesar de demandar um nível elevado de conhecimento, a construção de algoritmos complexos pode ser mais eficiente e rápida em termos de linhas de código. Isso se deve à maior flexibilidade na ordenação dos fatores, o que também facilita a manutenção posterior. Para o desenvolvimento da interação entre os recursos 3D, será utilizado o Blueprint.

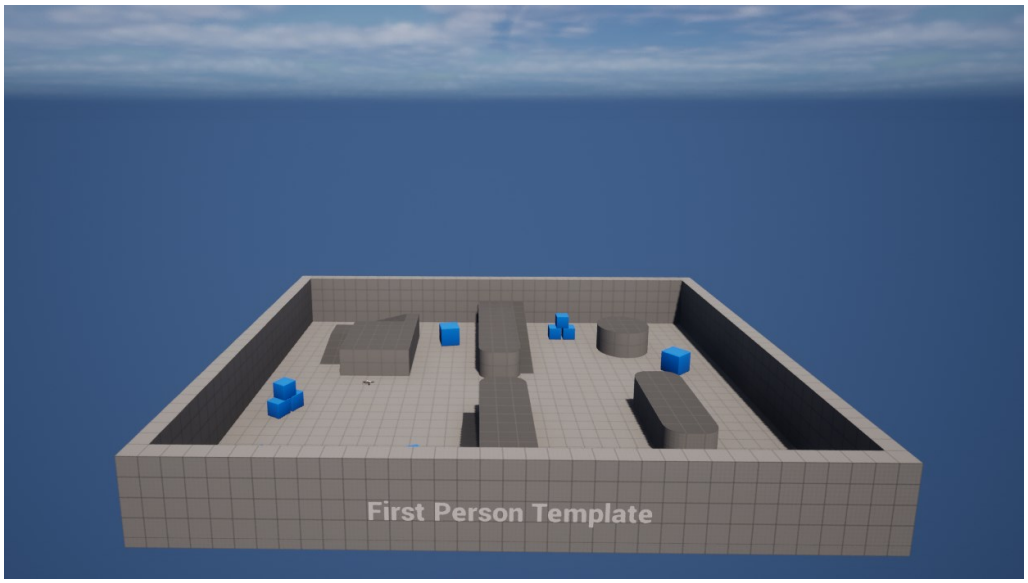
3.3 MONTAGEM DO SIMULADOR

A Unreal Engine se trata de um motor de jogo avançado e completo desenvolvido pela Epic Games. O motor de jogo pode ser considerado um sistema para criação e modificação de um ambiente 3D, facilitando e permitindo a criação de jogos, e no caso deste trabalho, simuladores imersivos. Lançado pela primeira vez em 1998, o Unreal Engine tem sido um dos principais produto do mercado de desenvolvimento de jogos, que permite aos desenvolvedores criar jogos de alta qualidade, além de facilitar o lançamento para outros dispositivos como consoles e dispositivos móveis. Com o passar dos anos, o Unreal Engine evoluiu para além dos jogos, sendo agora utilizado em simulações, visualizações arquitetônicas, produções cinematográficas e realidade virtual. A plataforma é conhecida por sua capacidade de renderização em tempo real, que produz gráficos de alta fidelidade e ambientes imersivos.

Essa ferramenta é altamente acessível, com uma versão gratuita disponível para pequenos desenvolvedores, e um modelo de licenciamento baseado em royalties para projetos comerciais de maior escala.

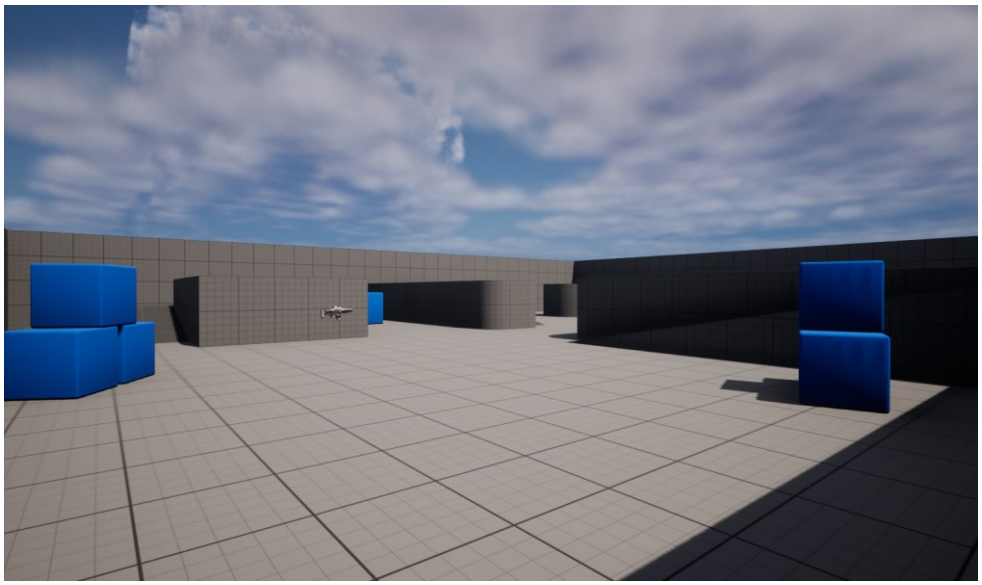
Após a modelagem feito pelo Blender, os ativos 3D serão importados para Unreal Engine. No Unreal, é possível montar as cenas, configurar a iluminação, adicionar efeitos especiais e criar a lógica de interação. A *engine* fornece vários *templates* padrão, e será selecionado o *template* de primeira pessoa, no qual a câmera é focada para representar os olhos de uma pessoa. Essa abordagem visa aumentar a percepção da simulação, permitindo que os olhos do participante visualizem todo o mundo 3D, simulando a visão do ser humano. A Figura 11, traz o mundo criado ao selecionar o *template* em primeira pessoa. A Figura 12, traz a visão do participante em primeira pessoa.

Figura 11 – *Template* padrão de primeira pessoa da Unreal Engine



Fonte: Imagem retirada do software pelo autor, 2023

Figura 12 – Visão do participante no mundo padrão



Fonte: Imagem retirada do software pelo autor, 2023

Ao selecionar o *template*, um novo mundo é criado com um personagem principal pré-configurado, e configurações de céu e iluminação natural já estabelecidas. O controle do personagem é possível utilizando os comandos W, A, S, D, além da capacidade de pular com a tecla de espaço. Embora o mundo seja modificado para atingir os objetivos deste trabalho, as configurações de movimento do personagem, bem como a iluminação natural permanecerão inalteradas. Os equipamentos serão importados um a um e posicionados conforme o layout representado na Figura 9. Em geral, os modelos 3D não possuem texturas ao serem importados do Blender para o Unreal Engine. Portanto, será aplicada uma configuração de material padrão fornecida pela própria *engine*, incorporando características metálicas aos componentes e aprimorando a qualidade dos ativos. Após testes, o material aço foi escolhido para todos os equipamentos.

A proposta prevê a interação com os seguintes equipamentos:

- Bomba para vazão de substrato
- Válvula para controle de vazão
- Trocador de controle para visualizar resultados
- Válvula de saída do biogás para visualizar resultados
- Válvula de saída do digestato para visualizar resultados

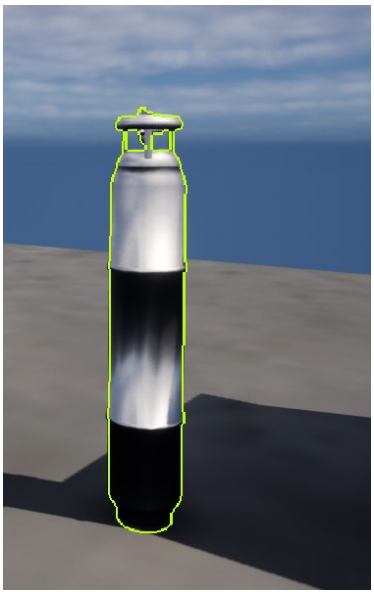
Além das interações com os objetos, também serão incluídas animações que permita visualizar a adição e remoção de líquido no biodigestor e a rotação do agitador. Serão incluídos efeitos sonoros para simular o movimento de válvula, ativação e desativação da bomba, e entrada do substrato no reator.

Toda questão programática que envolve as interações com o universo 3D, serão programadas via Blueprint, e somente os cálculos para solução de equacionamento químico de geração de biogás serão realizados no C++.

3.3.1 Interação com objetos 3D

No universo 3D, é desejável que as interações ocorram de maneira intuitiva, dispensando explicações detalhadas para sua execução. Nesse contexto, é crucial destacar objetos suscetíveis a ações, identificando-os para ativar ou desativar. Para alcançar esse objetivo, é configurado um material na Unreal Engine por meio do script em Blueprint. Este material é incorporado ao objeto quando um evento específico ocorre. O evento a ser configurado está relacionado com a proximidade do personagem ao objeto, gerando um contorno ao redor do objeto para destacá-lo sempre que o personagem se aproxima, indicando a disponibilidade de uma ação a ser realizada. Após a configuração desse material, é adicionado um recurso ao mundo chamado *PostProcess*, que é configurado adequadamente com o material criado. Posteriormente, é incorporado um evento ao objeto 3D que receberá a ação. Após a montagem dos scripts, obtemos o resultado conforme a Figura 13, e o usuário poderá interagir com a tecla “E”.

Figura 13 – Exemplo de cilindro de gás 3D com contorno de interação



Fonte: Imagem retirada do software pelo autor, 2023

A bomba centrífuga é responsável pela extração de líquido de um repositório e direcioná-la para um destino pré-determinado, no entanto seu uso é combinado com válvula, para controle de vazão uma vez que este só é responsável pela força efetuada para extração do líquido. O objetivo não é simular as condições físicas de uma bomba, e somente questões visuais e sonoras, além de ações que indicam que a vazão está ocorrendo durante a simulação.

A configuração inicial tem como objetivo permitir a interação com a bomba, proporcionando efeitos sonoros que indicam se está ligada ou desligada. Serão implementados dois recursos sonoros: o primeiro, representado como uma válvula, indicará quando a bomba foi acionada; o segundo consiste em um áudio contínuo que simula o barulho do motor da bomba. Esse áudio será interrompido apenas se o participante interagir novamente, ou seja, desligar a bomba. Vale ressaltar que a válvula só poderá ser utilizada após ligar a bomba. Se o participante a desligar com o Biodigestor em estado estacionário, o líquido retido no biorreator será reduzido a zero.

Após ligar a bomba, o usuário poderá manipular a válvula para controlar a vazão do líquido direcionado ao biorreator, medido em metros cúbicos (m^3). A válvula é exclusivamente responsável pelo controle de vazão do sistema. Ao interagir com ela, uma tela de controle de vazão será exibida, apresentando uma

barra que permite ao usuário ajustar a vazão dinamicamente usando o mouse ou inserindo um valor em m³ como texto.

A vazão de entrada da válvula tem valor mínimo e máximo, sendo 0 e 1000 $m^3.h^{-1}$ respectivamente, o usuário não poderá alterar o valor para fora deste limite. Ao adicionar o valor desejado à válvula e clicar no botão "*submit*", um som de válvula sendo girada será iniciado. A válvula girará para a direita se o novo fluxo for maior que o anterior, e para a esquerda caso contrário. Se os valores forem iguais, a posição permanece inalterada no mundo 3D. Durante esse processo de ajuste, uma mensagem de "carregando biodigestor" aparecerá acima da válvula, indicando que o uso dela estará indisponível temporariamente.

Após o líquido no biorreator atingir um nível estável, o processo de geração de biogás será iniciado, acompanhado por uma mensagem de "entrando em estado estacionário". Nesse momento, novamente, o uso da válvula estará temporariamente indisponível.

Ao interagir com o trocador de calor, deve-se abrir uma imagem ilustrativa do equipamento, mostrando os valores resultantes do cálculo de balanço de energia, para entrada e saída dos compostos quente e frio.

Logo após a interação com a válvula, o valor inserido assume o controle do nível de substrato dentro do reator, acompanhado por um efeito sonoro que simula a adição de líquido viscoso. A cor do líquido é diferenciada por um tom verde para representar um material orgânico, causando a sensação de resíduo em vez de algo palatável. A textura do fluido é animada para simular movimento, aumentando a percepção de realismo.

O nível do reator é ajustado de acordo com a vazão configurada na válvula. Aumenta se a vazão cresce e diminui se a vazão de entrada é reduzida. Se a redução alcançar o nível zero, o líquido desaparecerá. Para criar esse resultado visual, é necessário desenvolver uma animação que interaja com duas variáveis desse objeto. Caso contrário, o líquido aumentaria de forma estática.

O líquido no mundo 3D é representado por um cilindro, previamente configurado com localização e tamanho base para iniciar na mesma posição do biodigestor. Inicialmente, o líquido está oculto e aparece apenas quando necessário. Na Unreal Engine, o autor não encontrou a configuração para aumentar um objeto em uma única direção exclusiva em um determinado

eixo; pois, ao aumentar a escala, sempre aumentará em ambas as direções, na direção positiva e negativa. Se for necessário alterar o eixo Z, tanto a direção positiva quanto a negativa serão afetadas. Portanto, a alteração dinâmica apenas nessa variável causaria distorções no mundo 3D, prejudicando os detalhes da simulação.

O objetivo é aumentar o valor da escala do objeto junto com sua localização no eixo Z. Isso permite que ambos cresçam continuamente em conjunto, criando o efeito visual que indica o crescimento ou diminuição do volume dentro do reator. Quando a variável responsável pelo crescimento aumenta, a localização também aumenta, evitando que o crescimento ultrapasse os blocos 3D. Considerando os limites da vazão de entrada e buscando os valores para a posição ideal do fluido em seu limite máximo por meio de testes manuais para verificar um efeito visual razoável, desenvolve-se uma função linear para obter o valor da localização em função do fluxo de entrada. Isso é representado pela Equação 22, onde M é o coeficiente angular, podendo ser encontrado pela Equação 23, e b é o coeficiente linear determinado após a aplicação de valores conhecidos.

$$y = mx + b \tag{22}$$

$$m = \frac{\Delta y}{\Delta x} \tag{23}$$

Após determinar o valor final da localização por meio de testes, identificou-se que o valor da escala pode ser calculado pela Equação 24:

$$EscalaZ = 0.0125 Localização Z \tag{24}$$

É importante mencionar que os valores alterados são exclusivamente no eixo Z, uma vez que os demais eixos já estão pré-configurados.

A velocidade com que o nível cresce ou diminui pode ser facilmente configurada pelo software usando uma função chamada *LInterp*. Essa função permite a adição de valores em função de um delta t e uma velocidade de interpolação, além de receber um valor objetivo e o valor atual. A velocidade determina se o aumento ou diminuição será rápido ou lento.

Ao ativar a variável *tick* do objeto, o evento é chamado continuamente, enviando um valor de delta t a todo momento para o algoritmo. Todo o algoritmo presente nesse evento é executado, modificando assim o nível do reator até atingir o valor objetivo e passar para a etapa final do simulador.

Os cálculos do modelo para determinar a quantidade de biogás formado são implementados em C++, conforme descrito na seção 3.2, e sua implementação pode ser visualizado nos códigos dos Apêndices entre A e E. O biodigestor acessa os resultados desses cálculos e executa ações correspondentes para visualizar a quantidade de biogás gerado. O modelo proposto deste trabalho prevê uso do estado estacionário e estático, no entanto, para fins didáticos e representação adequada na simulação, será proposta uma Equação Diferencial Ordinária (EDO) de primeira ordem. Esta busca obter o valor de biogás em função do tempo, com parâmetros como valor de biogás fornecido, valor de entrada, valor de biogás inicial e tempo, regido pela Equação 25. Assim, o simulador mostrará a formação de biogás ao longo do tempo até atingir o estado estacionário.

$$y(t) = y0 + KM(1 - e^{-t/\tau}) \quad (25)$$

$$M = \Delta\text{ENTRADA} \quad (26)$$

$$K = \frac{\Delta y}{M} \quad (27)$$

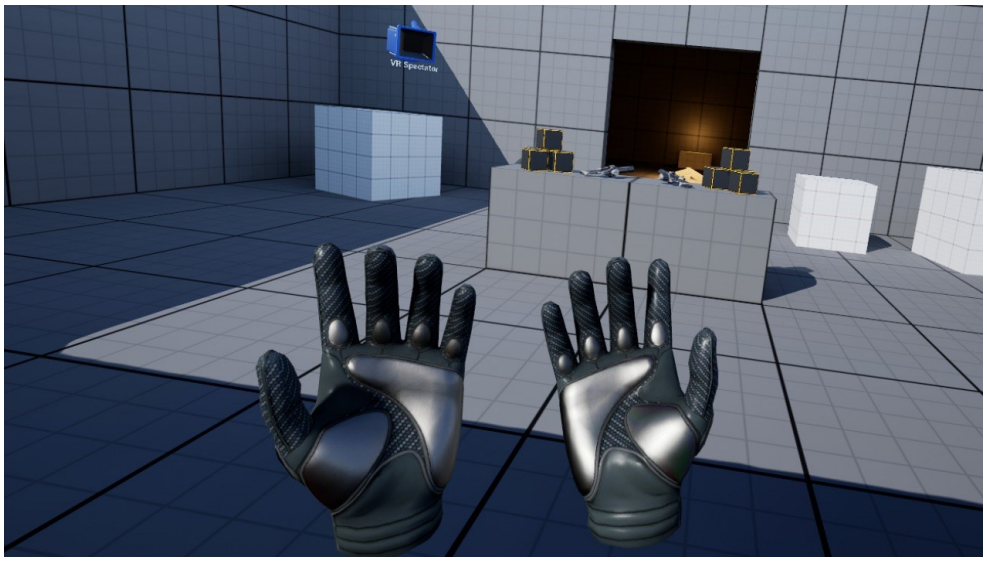
Uma janela se abre para o usuário assim que os cálculos são iniciados, mostrando o valor do biogás variando com o tempo no simulador. A variação desse tempo é facilmente ajustável pelo parâmetro de tempo do *tick* do objeto, que é um evento que executa os comandos em função do tempo real, controlando o delta t da variação. Isso possibilita configurar um valor razoável que simula o tempo para a geração de biogás, de acordo com a necessidade.

3.4 REALIDADE VIRTUAL (VR)

Inicialmente, o simulador será desenvolvido para a categoria de primeira pessoa, utilizando um *template* específico fornecido pela Unreal Engine. Nessa categoria, a entrada de dados ocorrerá por meio do teclado e mouse, permitindo

movimentação com as teclas A (esquerda), W (frente), S (atrás), D (direita), e controle da câmera através do mouse. Adicionalmente, foi introduzida a tecla “E” para interação com o ambiente. A Figura 14 ilustra o *template* padrão de VR, bem como a visão do usuário

Figura 14 – *Template* padrão do VR e visão do usuário



Fonte: Autor, 2023.

Todo o ambiente configurado neste formato será adaptado para a versão de realidade virtual. Alguns recursos serão ajustados para oferecer uma experiência mais imersiva e próxima da realidade. Essa versão específica destina-se ao uso de óculos de realidade virtual, proporcionando interações visuais e auditivas. Além disso, será possível simular a presença de mãos virtuais através de dois controles fornecidos pelo dispositivo tecnológico.

Dessa forma, a experiência proporcionada pelo simulador ultrapassará limites convencionais, permitindo que o usuário se aproxime do processo de produção de biogás. Ao explorar o funcionamento detalhado do sistema, o usuário poderá visualizar resultados gerados por cálculos que estimam a quantidade de biogás a ser produzida, com base na vazão de biomassa no sistema.

Serão implementadas mudanças no simulador, incluindo a introdução da visualização de painéis. No formato anterior, a tela era estática, permitindo interação com teclado e mouse. Na versão de realidade virtual (VR), a tela será

modelada como um objeto dinâmico no ambiente, ajustando-se à posição da câmera do participante e com tamanho apropriado. A interação ocorrerá através do controle do dispositivo, gerando um laser para indicar as seleções. Durante o desenvolvimento, a movimentação inicial será controlada pelo analógico dos óculos, podendo ser posteriormente adaptada para acompanhar o deslocamento do participante.

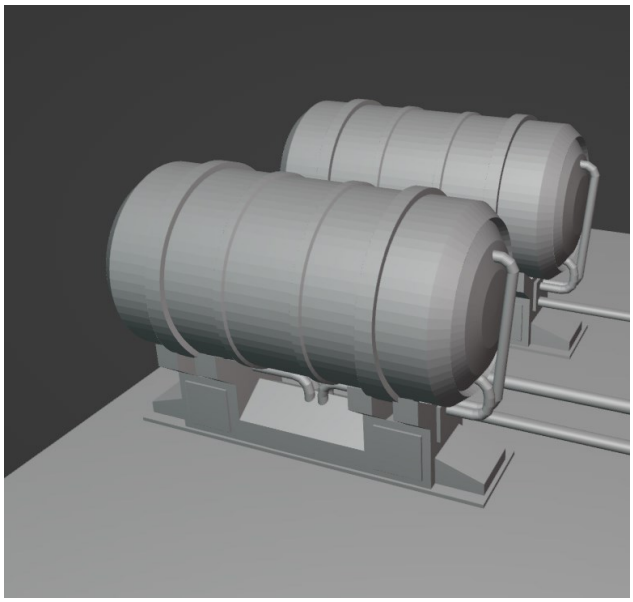
Ambos os modelos estarão disponíveis no simulador, sendo ativados conforme o dispositivo em uso. No caso de um computador, o modo de primeira pessoa será habilitado; enquanto, ao utilizar os óculos de realidade virtual, o modo VR será ativado. A Equação Diferencial Ordinária (EDO) da Equação 25 não será implementada na versão de Realidade Virtual (VR), isso se deve à complexidade do algoritmo de implementação, para o qual não haveria tempo suficiente até o final do projeto. Portanto, a visualização dos resultados será em uma versão estática, não sofrendo alterações com o decorrer do tempo.

4 RESULTADOS E DISCUSSÕES

4.1 RESULTADOS DA MODELAGEM 3D

A modelagem 3D tomou como base os equipamentos descritos na seção 3.1. Portanto, para cada equipamento obteve-se os resultados conforme a seguir. Os tanques foram modelados em 3Ds, a partir de modelos gratuitos e livres de direitos autorais conforme Figura 15. Por questão de praticidade, foram utilizados o mesmo para ambos, uma vez que a modelagem específica possui alta complexidade, e foge do escopo deste trabalho.

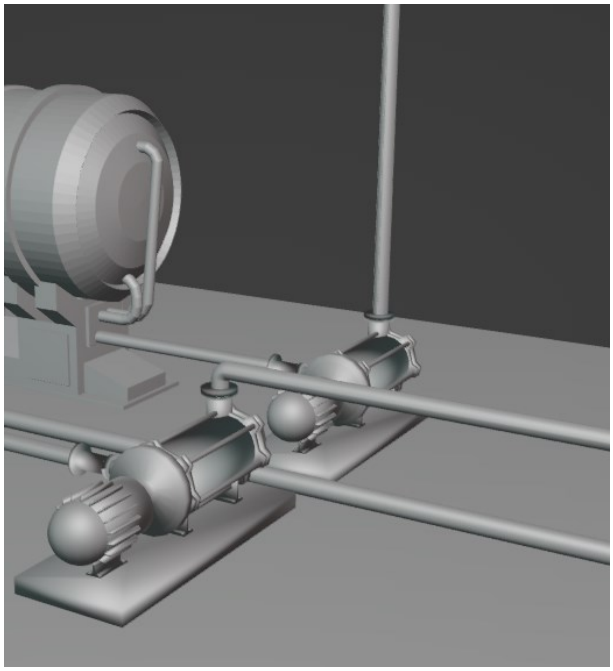
Figura 15 – Tanques de armazenamento de água e biomassa



Fonte: Autor, 2023

As bombas foram modeladas de acordo com a Figura 16, com as tubulações devidamente conectadas, com os tanques de armazenamento.

Figura 16 – Bombas e tubulações conectadas aos tanques

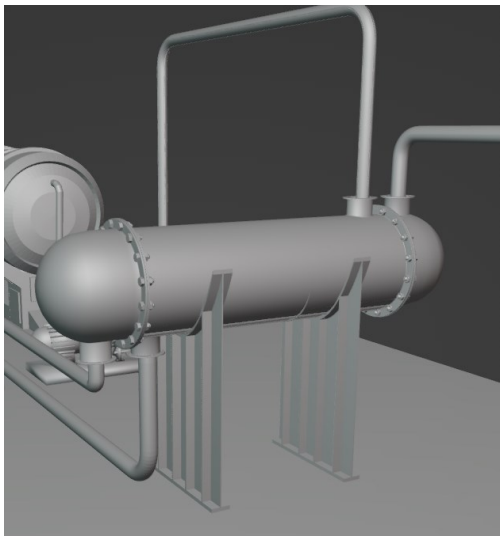


Fonte: Autor, 2023

O trocador de calor devidamente escolhido, foi modelado a partir de modelos gratuitos e livres de direitos autorais conforme Figura 17, o qual, a

tubulação da biomassa está conectada no casco e a água aquecida está conectada ao trocador de calor.

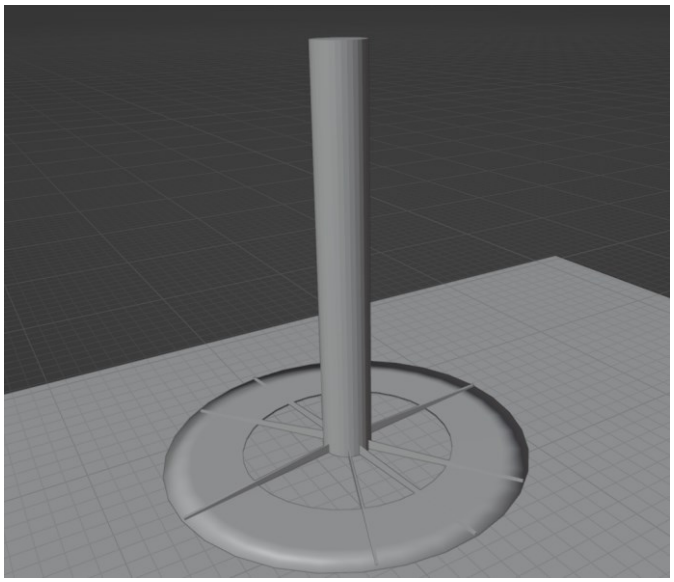
Figura 17 – Trocador de calor



Fonte: Autor, 2023

O agitador foi modelado separadamente e acoplado posteriormente ao biodigestor, conforme Figura 18.

Figura 18 – Agitador para biodigestor

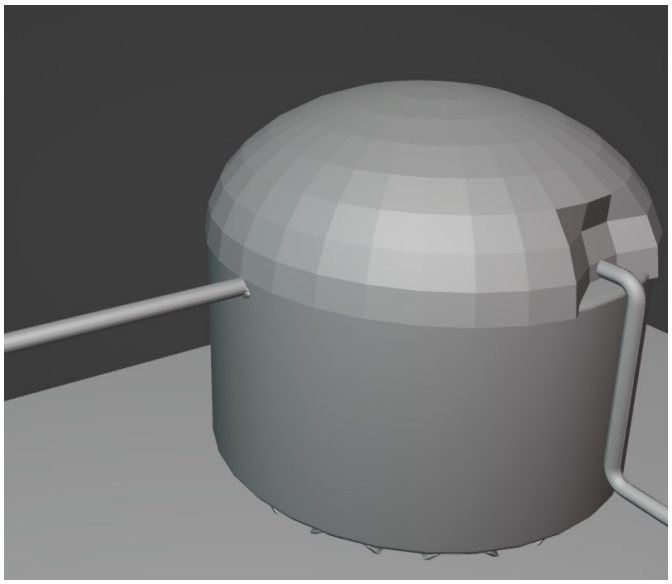


Fonte: Autor, 2023

O biodigestor foi modelado com base em um reator tipo de tanque agitado contínuo (CSTR). Na configuração, a biomassa é introduzida no tanque

pela parte superior, o biogás é retirado do topo, e o digestato é removido pelo fundo. A Figura 19 ilustra esse equipamento modelado.

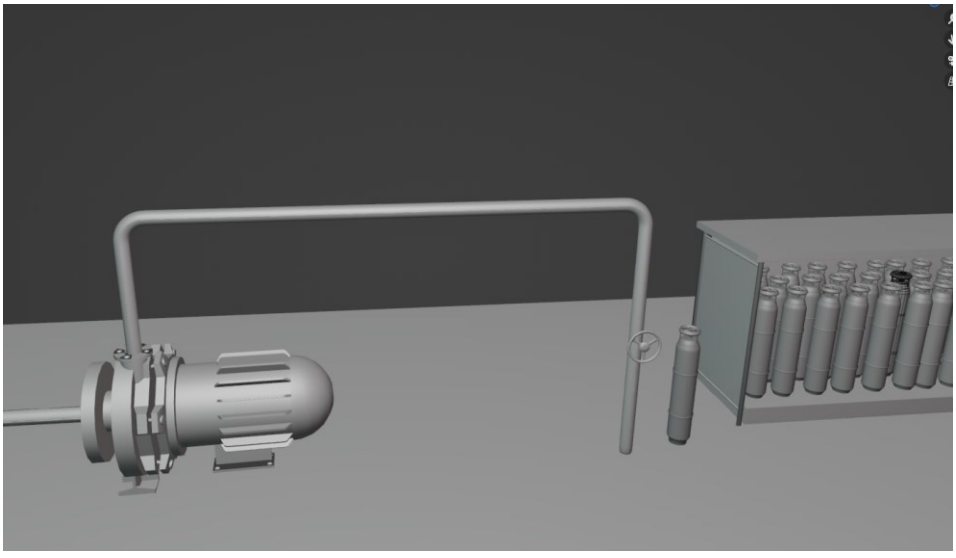
Figura 19 – Biodigestor



Fonte: Autor, 2023

A extração do biogás se dá a um compressor conectado ao biodigestor, via tubulação, o qual há uma válvula para retirada do gás e inserção em um cilindro de gás, conforme Figura 20.

Figura 20 – Compressor e válvula para retirada do biogás



Fonte: Autor, 2023

Os equipamentos modelados foram mensurados pelo próprio software Blender, e pode ser visualizado pela Tabela 7, com o intuito de fornecer um parâmetro comparativo e estimular a imaginação. Isso porque essas medidas não têm efeito no mundo virtual, já que as escalas podem ser alteradas apenas para criar a ilusão de objetos grandes ou pequenos em relação ao usuário que interage com o ambiente.

Tabela 7 – Medidas de equipamentos modelados

Equipamento	Altura (m)	Diâmetro (m)	Comprimento (m)	Largura (m)
Bombas	6,76	-	19,15	4,65
Casa de gás	14,89	-	41,85	10,70
Compressor	8,00	-	18,00	6,00
Tanques de armazenamento	-	16,63	30,72	-
Trocador de calor	-	8,22	34,52	-
Biodigestor	32,00	34,00	-	-
Agitador	13,36	12,59	-	-
Válvulas	-	2,45	-	-

Fonte: Autor, 2023

As medidas das tubulações que interligam os equipamentos estão dispostas na Tabela 8, seguindo a legenda das correntes da Figura 9.

Tabela 8 – Medidas de tubulações entre equipamentos

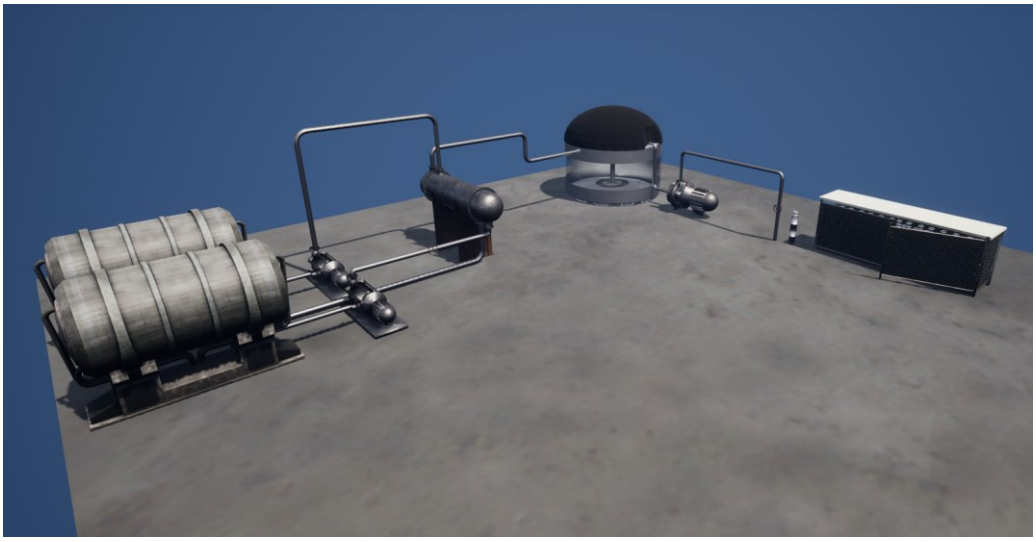
Corrente	Medida (m)
A-1	16,66
A-2	76,49
A-3	84,45
B-1	16,75
B-2	39,93
B-3	63,89
P-1	24,27
P-2	62,76

Fonte: Autor, 2023

4.2 RESULTADOS DO SIMULADOR 3D

Logo após a modelagem 3D, os equipamentos foram importados para o software Unreal Engine na versão 5.2, e foram disposto conforme Figura 21. O material aço metálico do próprio software, foi incorporado a cada um dos equipamento, de forma que os detalhes foram realçados.

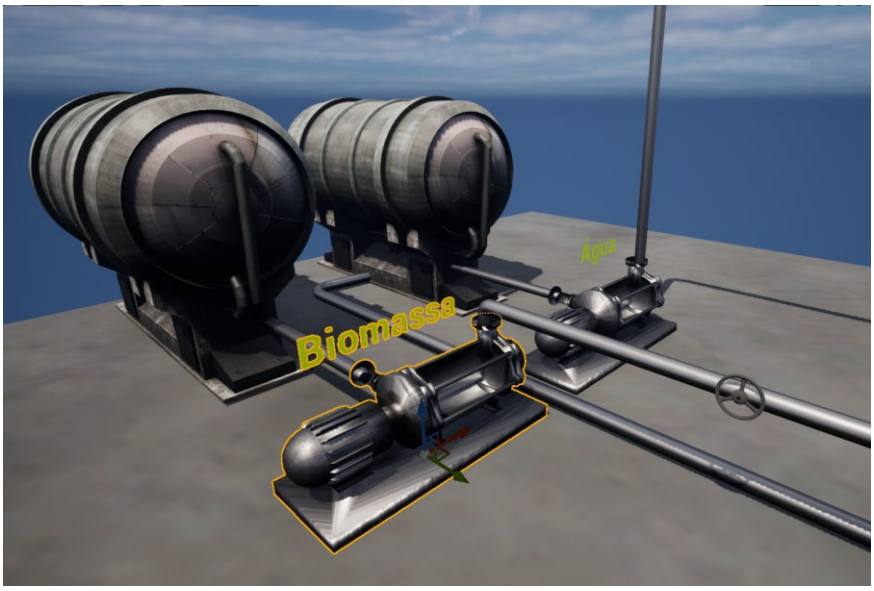
Figura 21 – Visão geral dos equipamentos montados no simulador



Fonte: Autor, 2023

A área das bombas e tanques podem ser visualizadas com mais detalhes na Figura 22, o qual há uma válvula para controle de vazão entre a bomba e o trocador de calor.

Figura 22 – Área das bombas e tanques de armazenamento



Fonte: Autor, 2023

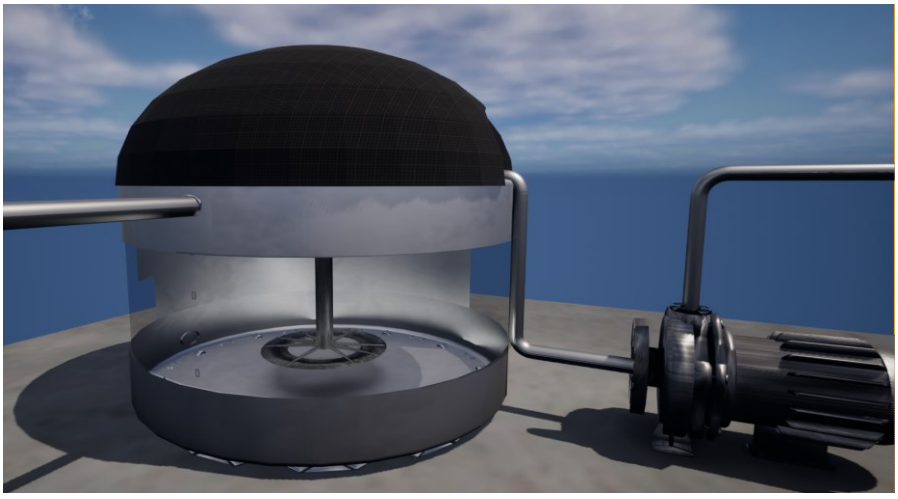
O trocador de calor casco-tubo, pode ser visualizado com mais detalhes na Figura 23.

Figura 23 – Área do trocador de calor

Fonte: Autor, 2023

A área do biodigestor, bem como o compressor de retirada de biogás, pode ser visualizada com mais detalhes na Figura 24. Observa-se o agitador interno, para mistura do digestato. Adicionou-se a película de vidro no biodigestor, com a intenção de visualizar internamente o que ocorre dentro do reator, no entanto, é importante reforçar que isso não é aplicável na realidade, tendo apenas fins didáticos para o contexto do simulador.

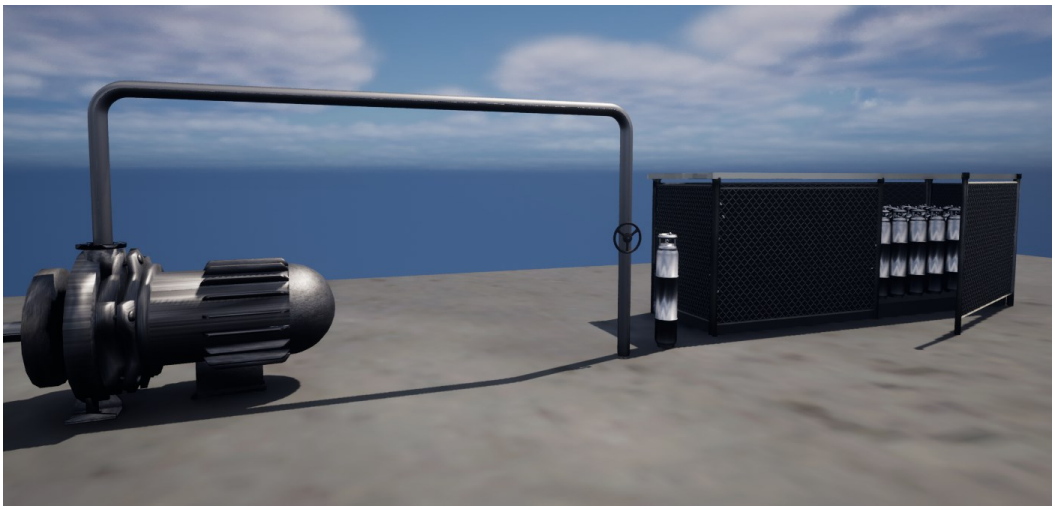
Figura 24 – Área do biodigestor



Fonte: Autor, 2023

O Gás será retirado à um cilindro e posteriormente transportado para para casa de gás, conforme Figura 25.

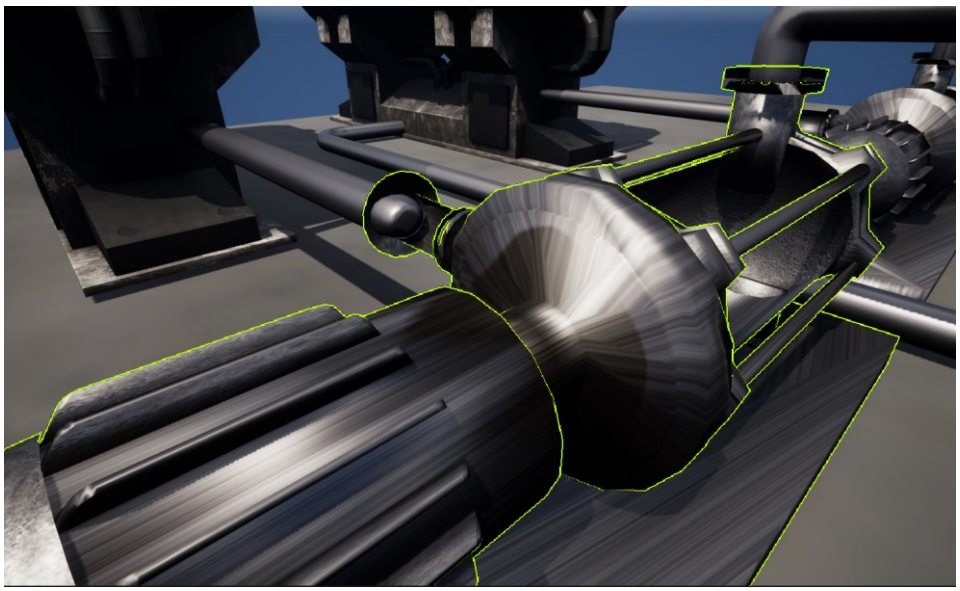
Figura 25 – Área da casa de gás, com válvula de retirada do biogás.



Fonte: Autor, 2023

As interações foram devidamentes desenvolvidas para indicar quando há interação ou não com o equipamento. Após ligar a bomba, pode-se escutar efeitos sonoros, como o som da válvula de ligar e desligar e um som contínuo do motor da bomba. A interação com a bomba pode ser visualizada conforme Figura 26.

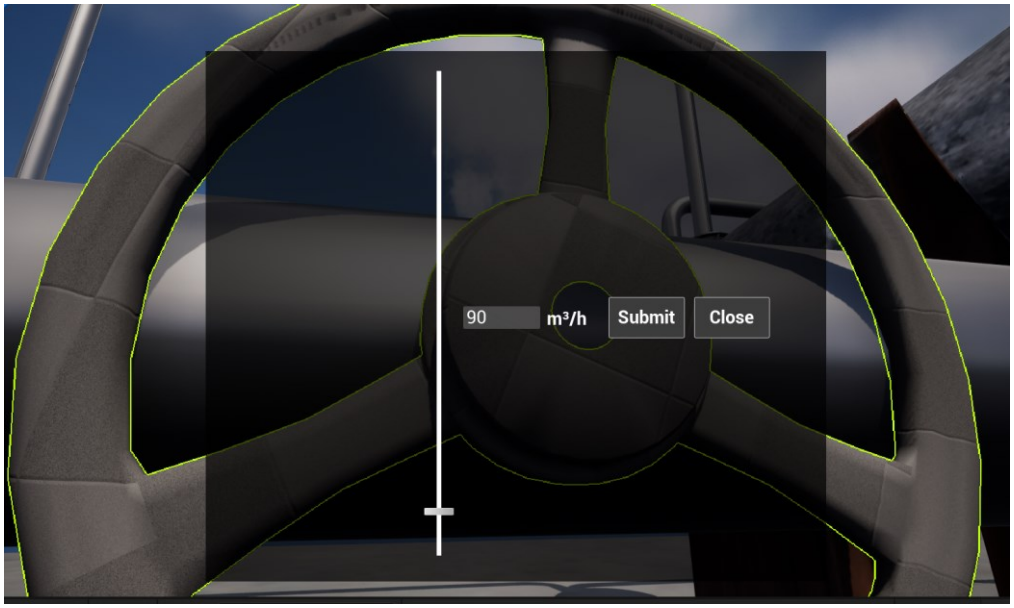
Figura 26 – Seleção de equipamento para interação



Fonte: Autor, 2023

Interação com a válvula de vazão para entrada da matéria-prima em direção ao trocador de calor, além da reprodução de efeitos sonoros simulando som de válvula em movimento. Após inserir o valor do fluxo de entrada no sistema, podendo alterar tanto pela barra de aumento, quanto digitando, possuindo valores máximos entre 0 e 1000 m³, sendo possível somente se a bomba estiver ligada. A Figura 27 ilustra esta interação.

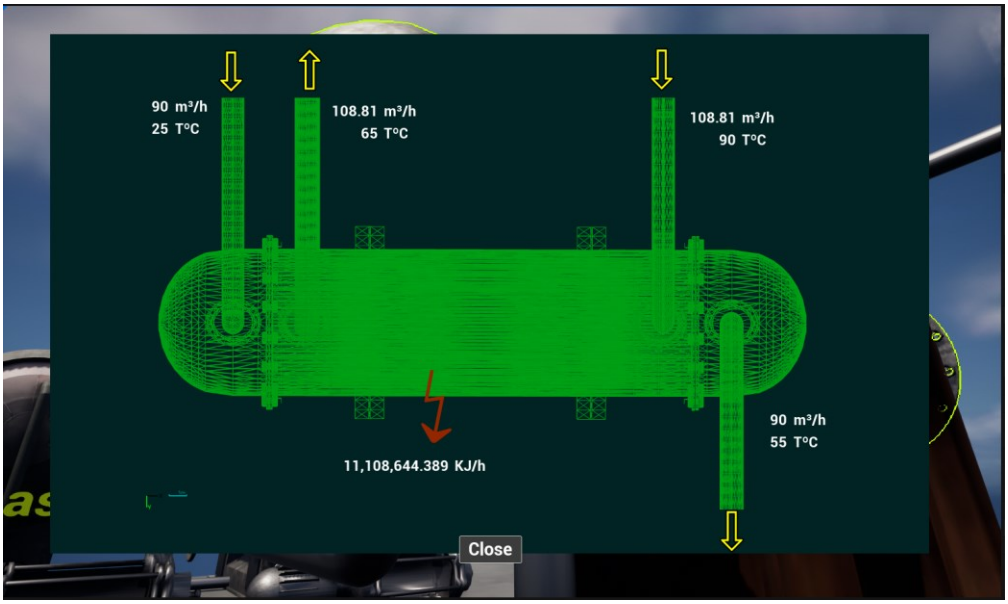
Figura 27 – Interação com a válvula de entrada de matéria-prima



Fonte: Autor, 2023

Após clicar em "*submit*", o próximo passo é o aquecimento da biomassa pelo trocador de calor, portanto, ao interagir com o trocador de calor, obtém-se o resultado do balanço de energia e massa, em suas respectivas entradas e saídas dos fluidos quente e frio, conforme Figura 28.

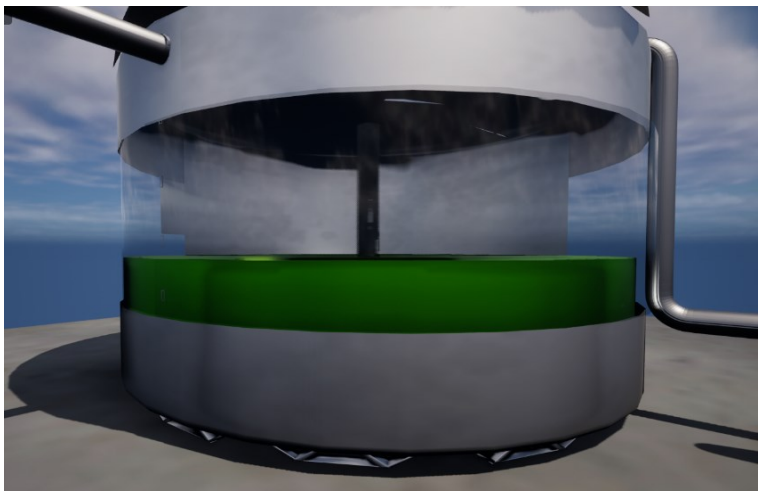
Figura 28 – Resultado da interação com o trocador de calor



Fonte: Autor, 2023

Logo após a saída do trocador de calor, já se inicia o processo de enchimento do reator, o qual o líquido possui animação aumentando e diminuindo conforme o valor de entrada da biomassa. Além do agitador estar ligado a todo momento. Essa animação pode parcialmente ser vista na Figura 29.

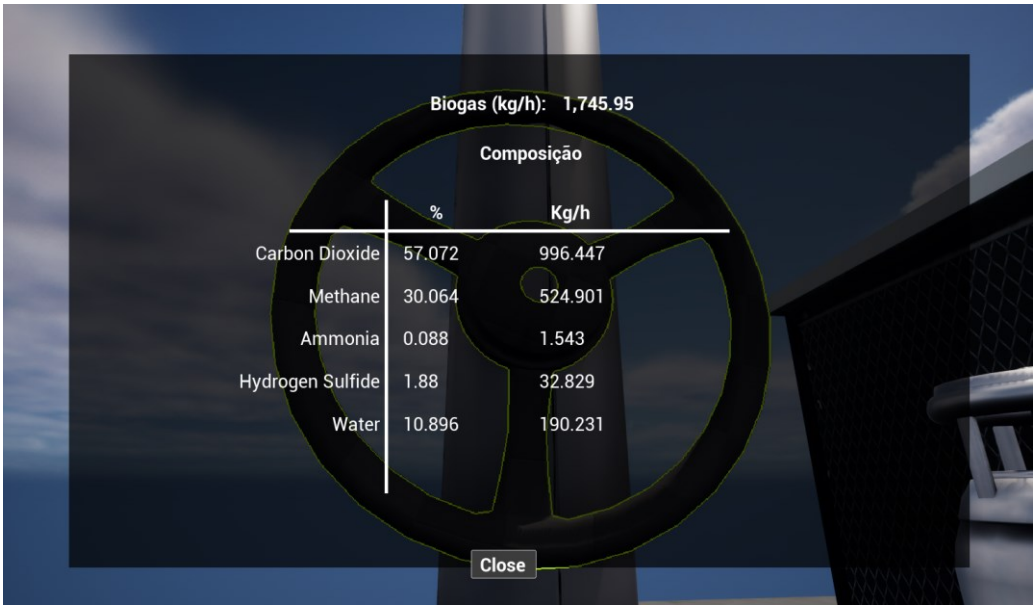
Figura 29 – Enchimento do biodigestor com a biomassa



Fonte: Autor, 2023

Por fim, obtém-se o resultado dos cálculos, podendo ser visualizados em forma de uma tabela ao interagir com as válvulas de saída de biogás e saída de digestato, conforme as Figuras 30 e 31, respectivamente.

Figura 30 – Resultado da geração de biogás



Fonte: Autor, 2023

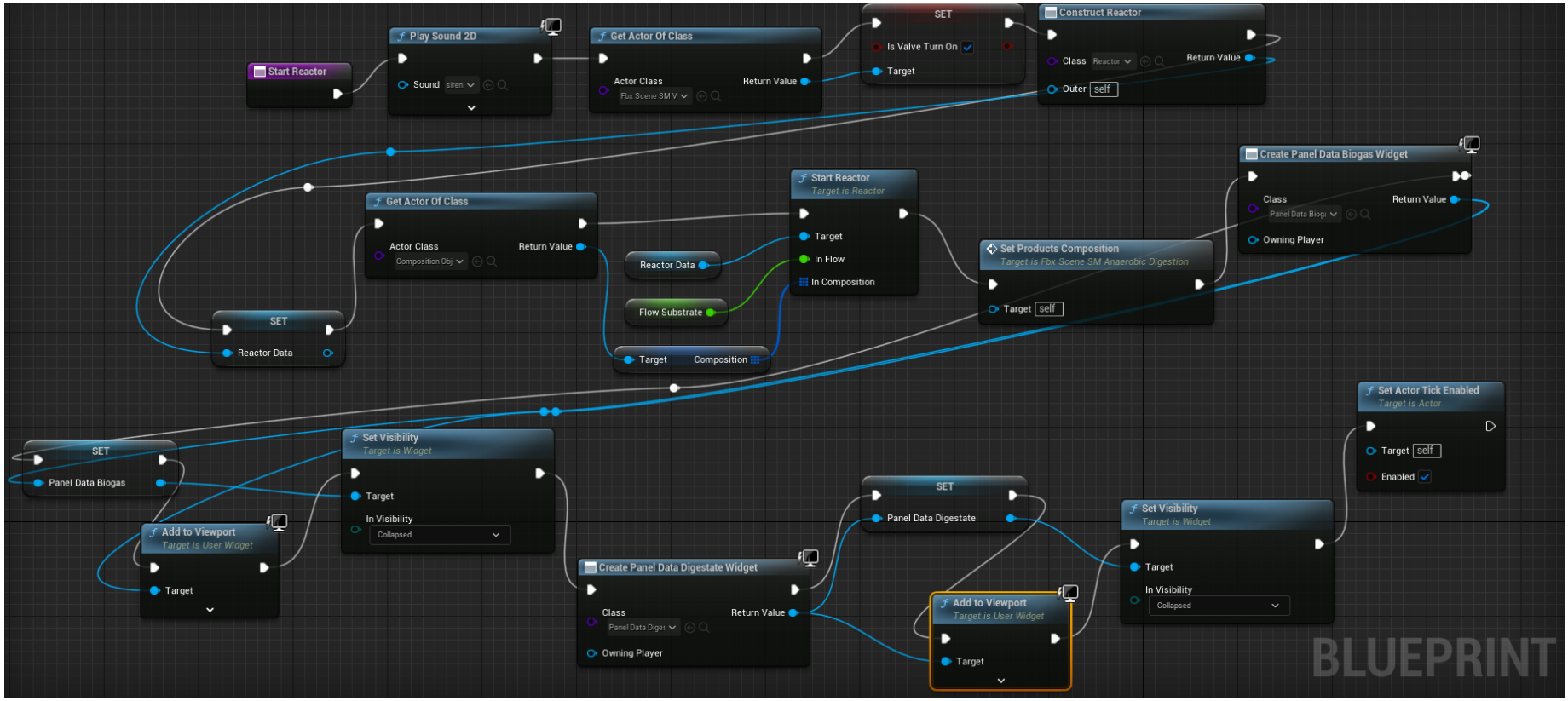
Figura 31 – Resultado da geração de digestato

Fonte: Autor, 2023

Os principais equipamentos, como o trocador de calor, biodigestor, bombas e válvulas, possuem funcionalidades implementadas em Script Blueprint. Esses scripts carregam códigos desenvolvidos em C++, conforme detalhado no apêndice deste trabalho, e executam diversas funções, como reproduzir som, alterar visualizações 3D e carregar outros scripts Blueprint desenvolvidos pelo autor.

Embora existam dezenas de Scripts *Blueprints* montados, cada um contendo funções específicas e extensas, não é possível apresentar todos detalhadamente neste trabalho. No entanto, na Figura 32, é possível observar o script Blueprint responsável por acessar o código em C++ para realizar os cálculos de simulação. Esse script não apenas traz os resultados, mas também os incorpora ao painel, conforme ilustrado nas Figuras 30 e 31.

Figura 32 – Script Blueprint para cálculos do reator na Unreal Engine



Fonte: Autor, 2023

Na Figura 32, é possível observar o script da Blueprint responsável por executar os principais códigos de cálculo do biodigestor, além de acionar outras funcionalidades do simulador. O script é iniciado com a ativação de uma sirene, indicando ao usuário que o processo de enchimento do biodigestor foi concluído. Em seguida, a classe desenvolvida em C++ (Apêndice E) que

contém os códigos de cálculo do reator é carregada, e a função "*Start Reactor*" é executada para iniciar os cálculos contidos nesse código. A partir desse nó, todos os resultados do simulador já estão carregados. Para carregar as variáveis que contêm todos os dados de composição e produtos formados, uma função em *Blueprint* é acionada: "*Set Products Composition*". A sequência final é composta por scripts desenvolvidos pelo autor para exibir os resultados na tela.

4.3 RESULTADOS QUANTITATIVOS

Os resultados dos cálculos para balanço de energia do trocador de calor, conforme Figura 28, são referentes ao valor de entrada de 90 m³/h de biomassa, e foram compiladas na Tabela 9.

Tabela 9 – Resultados obtidos para trocador de calor

Parâmetros	Biomassa	Água Aquecida
T entrada (°C)	25,00	90,00
T saída (°C)	55,00	65,00
Vazão (m³/h)	90,00	108,81
Calor (KJ/h)	$11,11.10^6$	

Fonte: Autor, 2023

Os resultados obtidos da geração de biogás e digestato pelo biodigestor, conforme Figuras 30 e 31, se referem a simulação para a entrada de 90 m³/h de biomassa no biodigestor, das quais os resultados do balanço de massa, podem ser visualizados na Tabela 10.

Tabela 10 – Balanço de mol e massa do biodigestor

Corrente	Vazão Mássica (kg/h)	Vazão Molar (kmol/h)
Entrada de Biomassa	90.000,09	4903,62
Saída de Biogás	1.745,95	66,97
Saída de Digestato	88.100,30	4.873,84

Fonte: Autor, 2023

Ao validar os valores, eles são divergentes ao obter o resultado do balanço de massa, no entanto vale ressaltar que a massa de microrganismos resultante do processo de biodigestão, devido ao crescimento microbiano, foi descartada das correntes de saída, sendo um dos prováveis motivos dessa

divergência.

As composições calculadas para biogás e digestato, foram compiladas para as Tabelas 11 e 12, respectivamente.

Tabela 11 – Resultado da composição do biogás

Compostos	Composição (%)	Vazão (kg/h)	Composição $\%_{mol}$	Vazão (kmol/h)
Dióxido de Carbono	57,072	996,447	33,807	22,642
Metano	30,064	524,901	48,855	32,720
Amônia	0,088	1,543	0,135	0,091
Sulfeto de Hidrogênio	1,880	32,829	1,438	0,963
Água	10,896	190,231	15,767	10,560

Fonte: Autor, 2023

Observa-se que em fração molar, obtém-se o resultado que representa 48%. Ao converter para base seca, tem-se o valor de 58% de metano, um resultado considerável, dentro do esperado para a biomassa em questão, reportado por Taiatele (2023).

Tabela 12 – Resultado da composição do digestato

Compostos	Composição (%)	Vazão (kg/h)	Composição $\%_{mol}$	Vazão (kmol/h)
Água	99,585	87.734,44	0,999	4.869,996
Proteína	0,372	328,083	<0,001	0,893
H	<0,001	0,420	<0,001	0,417
C	0,003	2,266	<0,001	0,189
N	0,030	26,046	0,004	18,595
S	0,003	2,381	<0,001	0,074
O	0,008	6,660	<0,001	0,416

Fonte: Autor, 2023

Onde, os elementos H, C, N, S e O são derivados de compostos desconhecidos oriundos de componentes gasosos da corrente de biogás, que não se mantiveram gasosos, contribuindo então para a composição de digestato.

Taiatele (2023), realizou os cálculos de simulação utilizando o Aspen Plus®, e um modelo de Excel criado pelo próprio autor. Este trabalho focou na reprodução do modelo em Excel, e obteve os seguintes resultados, de forma comparativa, sendo observados na Tabela 13.

Tabela 13 – Comparação de resultados obtidos por outro autor

Modelo	Produto total (kmol)	Corrente de biogás (kmol)	Corrente de digestato (kmol)	%molar metano
Aspen Plus	4.745,24	64,56	4680,68	50,14
Excel	4.941,95	67,74	4874,20	48,32
Simulador 3D	4.940,81	66,97	4873,84	48,85

Fonte: Elaborado pelo autor com base em Taiatele, 2023.

Com base nos resultados obtidos, observa-se que o simulador 3D demonstrou desempenho satisfatório na reprodução dos cálculos de geração de biogás. Os valores gerados pelo simulador são notavelmente próximos aos alcançados pelo autor por meio do Excel. Vale ressaltar que os cálculos foram realizados pelo simulador, o qual se baseou nas formulações presentes no Excel do autor. Essa concordância entre os resultados do simulador 3D e os cálculos originais sugere que a ferramenta é eficaz na simulação do processo de geração de biogás, validando assim sua capacidade de reproduzir fielmente os cálculos realizados no ambiente do Excel pelo autor.

4.4 RESULTADOS DO MODO REALIDADE VIRTUAL

Conforme descrito na seção 3.4, o simulador foi todo reaproveitado para funcionar para uma versão de realidade virtual (VR). Assim obteve-se resultado satisfatório, sendo possível interagir com o ambiente através desse novo dispositivo. A movimentação se dá pelo controle do próprio VR, no entanto, há a possibilidade de alterar para acompanhar os passos do usuário, em uma futura implementação.

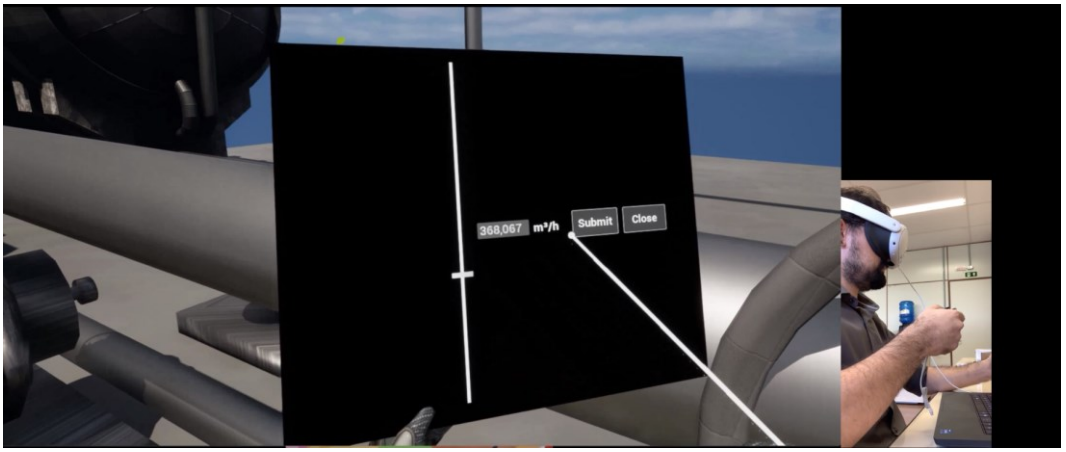
Observa-se na Figura 33, ao interagir com a bomba, a animação de seleção não está disponível, uma vez que esta configuração presente no modo primeira pessoa, não foi incorporado na versão VR, sendo necessário investigar a causa desse problema.

Figura 33 – Interação com a bomba na versão VR

Fonte: Autor, 2023

Ao interagir com a válvula, a mesma tela é replicada, porém, de forma dinâmica, acompanhando a movimentação da mão esquerda, e sendo possível a alteração dos valores com a mão direita, conforme Figura 34.

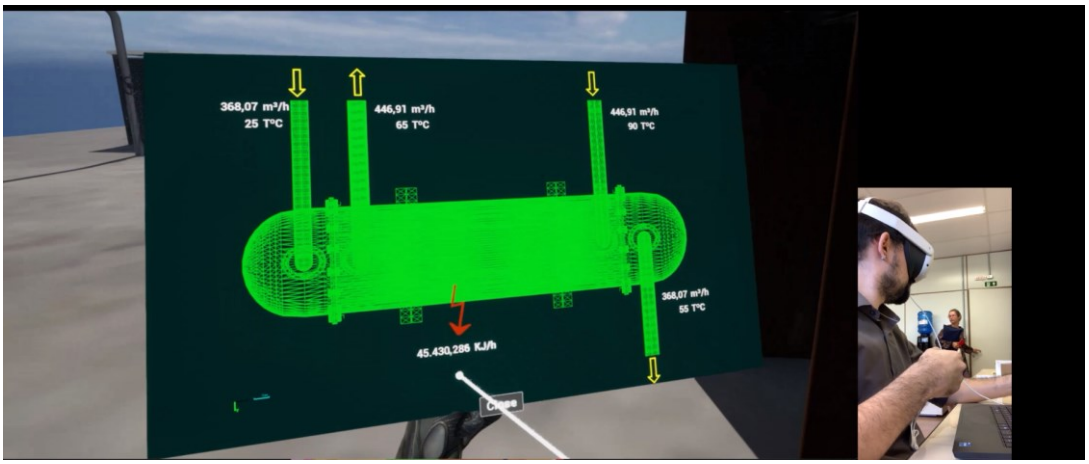
Figura 34 – Interação com a válvula na versão VR



Fonte: Autor, 2023

O mesmo comportamento é replicado para a interação com o trocador de calor, ao obter o painel de resultados do equipamento, conforme Figura 35.

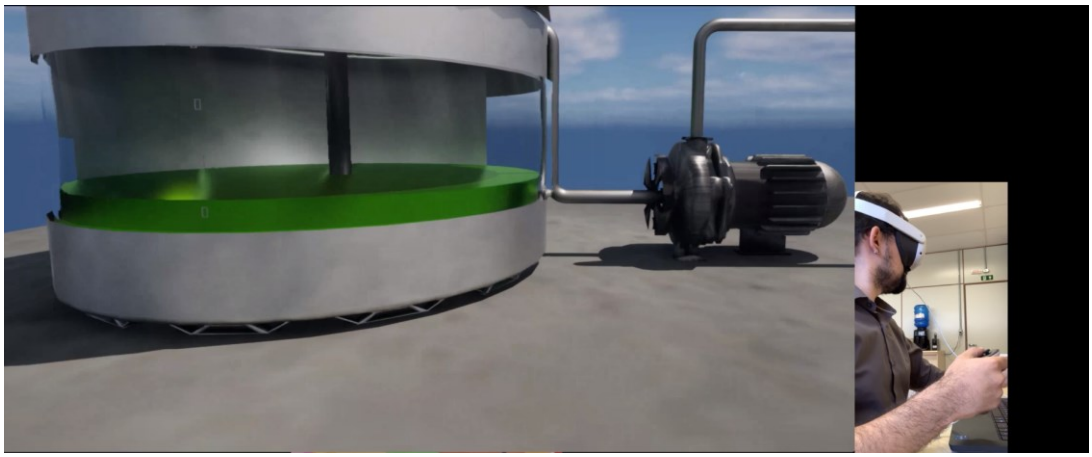
Figura 35 – Interação com o trocador de calor na versão VR



Fonte: Autor, 2023

A movimentação do fluido dentro do biodigestor, representando seu preenchimento permanece da mesma forma conforme em primeira pessoa, conforme Figura 36.

Figura 36 – Visualização do preenchimento do biodigestor na versão VR



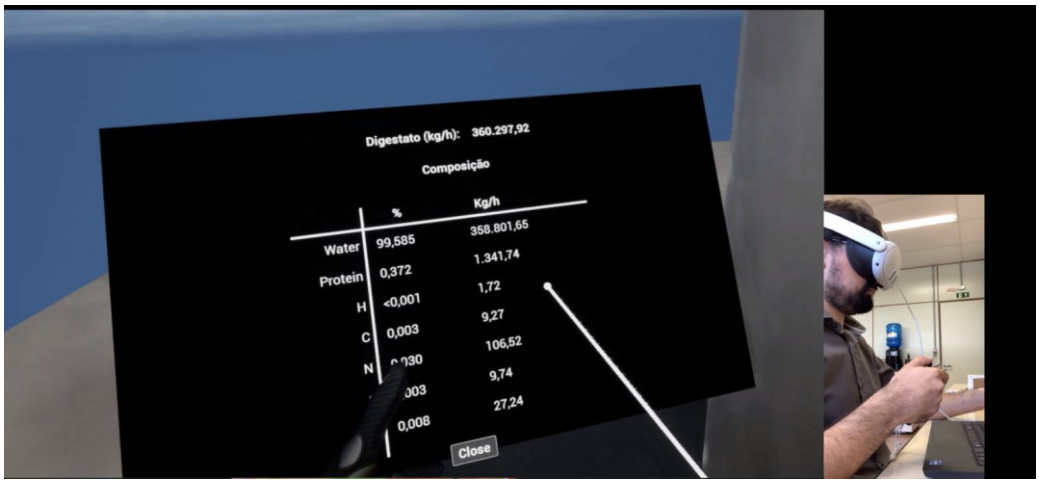
Fonte: Autor, 2023

Os resultados do biogás e digestato, são visualizados após o contato com as válvulas de saída, repetindo o mesmo comportamento da primeira válvula e do trocador de calor, obtendo os mesmos resultados para a versão de primeira pessoa, caso sejam replicados. A visualização do painel pode ser observada para biogás e digestato, conforme Figuras 37 e 38, respectivamente.

Figura 37 – Resultados do biogás na versão VR

Fonte: Autor, 2023

Figura 38 – Resultados do digestato na versão VR



Fonte: Autor, 2023

5 CONCLUSÕES

O simulador imersivo desenvolvido exibe uma performance considerável, fornecendo uma reprodução eficaz e precisa do processo de biodigestão anaeróbia de efluentes oriundos de fecularias de mandioca. Os cálculos relacionados à produção de biogás referem-se a um modelo teórico, onde quantificamos os resultados da formação de subprodutos por meio de quinze reações bioquímicas distintas. Essas reações abrangem as etapas de hidrólise, acidogênese, acetogênese e metanogênese.

Adicionalmente, o simulador contém oito elementos tridimensionais modelados pelo software Blender, que compõem o cenário virtual e facilitam a visualização do processo de biodigestão. Entre esses elementos, incluem-se bombas, um biodigestor, um trocador de calor, válvulas, tanques de armazenamento e tubulações. Ele oferece interatividade, permitindo que os usuários perturbem o sistema e analisem em tempo real os resultados. Além de cumprir esse propósito, o simulador tem o potencial de enriquecer a experiência daqueles interessados no processo de geração de biogás. Ao final, obteve-se 48% de metano na corrente de biogás, em base molar, valor satisfatório para a biomassa escolhido. O simulador 3D suporta uma versão de realidade virtual (VR) com resultados significativos, obtendo desempenho semelhante ao modo de primeira pessoa, embora com alguns comportamentos distintos, como já descrito anteriormente. No entanto, essas diferenças não comprometem a experiência de imersão.

Contudo, é importante destacar que há espaço para melhorias significativas. Primeiro, o modelo de cálculo de biogás não prevê modelos termodinâmicos e cinéticos, uma vez que parte do pressuposto de que todas as reações vão ocorrer dentro do esperado. Em relação a troca térmica, atualmente, a água aquecida é derivada de um tanque, o que não representa fielmente a prática real. Uma implementação futura poderia incorporar um aquecedor de água, como uma caldeira, para maior precisão. Além disso, o processo de purificação do biogás não foi abordado, apresentando apenas o resultado em sua forma bruta. Esses aprimoramentos potenciais podem tornar o simulador ainda mais robusto e representativo. No que diz respeito aos elementos 3D, é

possível aprimorá-los por meio de uma modelagem mais detalhada e a adição de pinturas realistas. Isso pode ser alcançado utilizando softwares específicos, como o 3D Painter, dedicados a essa finalidade.

Por fim, o simulador é capaz de acomodar uma ampla gama de valores de composição relacionados à biomassa escolhida. No entanto, vale ressaltar, que a precisão da simulação pode ser comprometida ao optar por outro tipo de biomassa, caracterizado por elementos distintos não previstos pelo sistema. Isso se deve à limitação do simulador em antecipar reações químicas envolvendo uma variedade de componentes não especificados. Assim, a ferramenta de simulação se destaca ao ser aplicada em Águas Residuárias de Mandioca, mostrando-se flexível em relação aos diversos valores de composição que podem ser inseridos para esse tipo específico de biomassa.

REFERÊNCIAS

ALQURAISH, M. Modeling and Simulation of Manufacturing Processes and Systems: Overview of Tools, Challenges, and Future Opportunities. **Engineering, Technology & Applied Science Research**, Greece, v. 12, n. 6, p. 9779–9786, 2022.

AMANHA. Disponível em: https://amanha.com.br/categoria/negocios-do-sul1/bndes-apoia-projeto-de-planta-de-biogas-no-parana. Acesso em 25 nov. 2023.

ARAUJO, I. R. C. *et al*. Methane production from cassava starch wastewater in packed-bed reactor and continuous flow. **Engenharia Agrícola**, [*S.l.*], v. 38, n. 2, p. 270-276, abr. 2018.

BANKS, C. J.; HEAVEN, S. Optimisation of biogas yields from anaerobic digestion by feedstock type. **The Biogas Handbook**, [*S.l.*], p. 131-165, 2013.

BLENDER (org.). **A história do Blender**. Disponível em: https://docs.blender.org/manual/pt/2.79/getting_started/about/history.html. Acesso em: 15 nov. 2023.

CREMONEZ, P. A. *et al*. Two-Stage anaerobic digestion in agroindustrial waste treatment: a review. **Journal Of Environmental Management**, [*S.l.*], v. 281, p. 111854, mar. 2021.

CRUZ, I. A. *et al*. Valorization of cassava residues for biogas production in Brazil based on the circular economy: an updated and comprehensive review. **Cleaner Engineering And Technology**, [*S.l.*], v. 4, p. 100196, out. 2021.

EBOMBAS. Disponível em: https://www.ebombas.com.br/bomba-centrifuga-schneider-me-in-1640n-4-cv-220-380-440-760v-trifasico?search=87230347-00&gad_source=1&gclid=CjwKCAiA9ourBhAVEiwA3L5RFjsoud2QFbdrxGtPmrh8K_AjDTe02e8OW836RrxrDa_9QhF8THdBhRoCpAIQAvD_BwE. Acesso em 20 nov. 2023.

INDUSTRIALIS. Disponível em: https://loja.industrialis.com.br/compressor-radial-460kw-613cv-trifasico?utm_source=Site&utm_medium=GoogleMerchant&utm_campaign=GoogleMerchant&gclid=CjwKCAiA9ourBhAVEiwA3L5RFsxO_fd6NRZuYeYQ-K9I9jkTI1QNCwHCQXbfNmEnK7fug5S0VllhIhoCAZQQAvD_BwE. Acesso em 20 nov. 2023.

KUMAR, V. V. *et al*. Virtual reality in chemical and biochemical engineering education and training. **Education For Chemical Engineers**, [*S.l.*], v. 36, p. 143-153, jul. 2021.

OVIVOWATER. Disponível em: https://www.ovivowater.com/en/product/anaerobic-sludge-mixers/ Acesso em 20 nov. 2023.

RAJENDRAN, K. *et al*. A novel process simulation model (PSM) for anaerobic digestion using Aspen Plus. **Bioresource Technology**, [*S.l.*], v. 168, p. 7-13, set. 2014.

SÁNCHEZ, A. S. *et al*. Waste bio-refineries for the cassava starch industry: new trends and review of alternatives. **Renewable And Sustainable Energy Reviews**, [*S.l.*], v. 73, p. 1265-1275, jun. 2017.

SERPA, Y. R. *et al*. An interactive simulation-based game of a manufacturing process in heavy industry. **Entertainment Computing**, [*S.l.*], v. 34, p. 100343, maio 2020.

SILVA, V. M. C. *et al*. Study of brain waves in the learning process in virtual reality. **Caderno Pedagógico**, *[S. l.]*, v. 19, n. 1, 2022.

SOLMAZ, S.; KESTER, L.; VAN GERVEN, T. An immersive virtual reality learning environment with CFD simulations: Unveiling the Virtual Garage concept. **Education and Information Technologies**, [*S.l.*], p. 1-34, mar. 2023.

TAIATELE JUNIOR, I. **Recuperação energética do biogás gerado a partir de água residuária de fecularias de mandioca: uma análise de viabilidade técnico-econômica e ambiental**. 2023. 143 p. Tese (Doutorado em Engenharia Civil) - Universidade Estadual de Londrina, Londrina, 2023.

TANQUESEBOMBAS. Disponível em: https://tanquesebombas.com.br/produto/tanque-aereo-estacionario-horizontal-novo-5000-litros. Acesso em 20 nov. 2023.

TERMOTEK. Disponível em: https://www.termotek.com.br/trocador-calor-tipo-casco-tubo. Acesso em 20 nov. 2023.

THE ENGINEERING TOOLBOX. Water - Specific Heat vs. Temperature. [online], 2004. Disponível em: https://www.engineeringtoolbox.com/specific-heat-capacity-water-d_660.html. Acesso em: 25 nov. 2023. Base de dados.

TOCU, N.A *et al*. The impact of virtual reality simulators in manufacturing industry. *In:* International Conference on Education and New Learning Technologies, 12., 2020, Online Conference. **Proceedings EDULEARN20**. Online: IATED, 2020, p. 3084-3093. Disponível em: https://doi.org/10.21125/edulearn.2020.0905. Acesso em: 10 nov. 2023.

TURNER, C. J.; GARN, W. Next generation DES simulation: a research agenda for human centric manufacturing systems. **Journal Of Industrial Information Integration**, [*S.l.*], v. 28, p. 100354, jul. 2022.

UNREAL ENGINE. **Unreal Engine 5**. Disponível em:

https://www.unrealengine.com/pt-BR/unreal-engine-5/. Acesso em: 26 nov. 2023.

VALMEC. Disponível em: https://valmec.com.br/produto/valvula-guilhotinafaca-passante-curta-210/. Acesso em 20 nov. 2023.

C++. *In*: WIKIPÉDIA: Simple English. [São Francisco, CA: Fundação Wikimedia], 2023. Disponível em: https://simple.wikipedia.org/wiki/C%2B%2B. Acesso em: 15 jun. 2023.

ZAYTSEV, Egor. **Blueprints vs Programming**. [*S.l.*], 16 jun. 2023. LinkedIn: Egor Zaytsev @egor-zaytsev. Disponível em: https://www.linkedin.com/pulse/blueprints-vs-programming-egor-zaytsev/. Acesso em: 17 nov. 2023

APÊNDICE A – CÓDIGO-FONTE PARA CRIAÇÃO DE COMPOSTO E MISTURA

Arquivo Compound.h

```
#pragma once

#include "CoreMinimal.h"
#include "UObject/NoExportTypes.h"
#include "Composition.h"
#include "Compound.generated.h"

USTRUCT(BlueprintType)
struct FGasLiquid
{
    GENERATED_BODY()

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    double LiquidFraction;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    double GasFraction;

    bool bIsEmpty;
    bool bIsNotEmpty;

    FGasLiquid() {}

    FGasLiquid(const double InLiquidFraction) :
LiquidFraction(InLiquidFraction)
    {
        GasFraction = 1 - InLiquidFraction;
        bIsNotEmpty = true;

    }

    FGasLiquid(const double InLiquidFraction, const double InGasFraction) :
LiquidFraction(InLiquidFraction), GasFraction(InGasFraction)
    {
        if (LiquidFraction == 0 || GasFraction == 0)
        {
            bIsEmpty = true;
        }
        else
        {
            bIsNotEmpty = true;
        }
    }

};

USTRUCT(BlueprintType)
struct FCompositionComponent
{
    GENERATED_BODY()

    UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category = "Compound")
    FString Name;

    UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category = "Compound")
```

```
    double FlowMass;

    UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category = "Compound")
    double FlowMolar;

    UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category = "Compound")
    double FlowVolume;

    UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category = "Compound")
    double Percent;

    FCompositionComponent() {}
    FCompositionComponent(FString inName, double inFlowMass, double
inFlowMolar, double inPercent )
        : Name(inName), FlowMass(inFlowMass), FlowMolar(inFlowMolar),
Percent(inPercent)
    {}

};

UCLASS(Blueprintable)
class MYPROJECT2_API UCompound : public UObject
{
        GENERATED_BODY()

public:

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    int32 Id;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    FString Name;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    FString Formula;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    FString CasNumber;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    FString CompType;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    double WeightMolar;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    double ConcInFlow = 0;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    double FlowMolar = 0;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    double GasFlowMolar = 0;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    double LiquidFlowMolar = 0;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    double Mass = 0;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    double GasMass = 0;
```

```
    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    double LiquidMass = 0;

    UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category = "Compound")
    double HeatCapacity = 0;

    UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category = "Compound")
    double Temperature = 0;

    UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category = "Compound")
    double Density = 0;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    double FlowVolume;


    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Compound")
    FGasLiquid FractionGasLiquid = FGasLiquid(0, 0);



    UCompound(){}

    UFUNCTION(BlueprintCallable, Category = "Compound")
    void Init(int32 inId, FString inName, FString inFormula, FString
inCasNumber, FString inCompType, double inWeightMolar);

    UFUNCTION(BlueprintCallable, Category = "Compound")
    void CalculateMass();

    UFUNCTION(BlueprintCallable, Category = "Compound")
    void PrintProperties();

    UFUNCTION(BlueprintCallable, Category = "Compound")
    void CalculateMassFromVolume();

    UFUNCTION(BlueprintCallable, Category = "Compound")
    void CalculateFlowMolar();

    UFUNCTION(BlueprintCallable, Category = "Compound")
    void CalculateVolumeFromMass();
};


UCLASS()
class MYPROJECT2_API  UMixture : public UCompound
{
    GENERATED_BODY()

public:
    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Mixture")
    TArray<UCompound*> Compounds;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Mixture")
    FString StateOfMatter;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Mixture")
    bool bIsLiquid;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Mixture")
    bool bIsGas;
```

```
    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Mixture")
    TArray<FCompositionComponent> Composition;


    UMixture(){}

    void Init(FString inName, FString inStateOfMatter, TArray<UCompound*>
inCompounds);

    UFUNCTION(BlueprintCallable, Category = "Mixture")
    void CalculateProperties();

    UFUNCTION(BlueprintCallable, Category = "Mixture")
    void CalcWeightMolar();

    UFUNCTION(BlueprintCallable, Category = "Mixture")
    void CalcFlowMolar();

    UFUNCTION(BlueprintCallable, Category = "Mixture")
    void SetComposition();

    UFUNCTION(BlueprintCallable, Category = "Mixture")
    TArray<FCompositionComponent> SetCompositionStructure();

    UFUNCTION(BlueprintCallable, Category = "Mixture")
    void SetConcentration(TArray<FCompositionData> inComposition, double
inFlowIn);

    UFUNCTION(BlueprintCallable, Category = "Mixture")
    UCompound* GetCompound(FString inName);

    void PrintProperties();
};
```

Arquivo Compound.cpp

```
#include "Compound.h"

void UCompound::Init(int32 inId, FString inName, FString inFormula, FString
inCasNumber, FString inCompType, double inWeightMolar) {
    this->Id = inId;
    this->Name = inName;
    this->Formula = inFormula;
    this->CasNumber = inCasNumber;
    this->CompType = inCompType;
    this->WeightMolar = inWeightMolar;
}

void UCompound::CalculateMassFromVolume() {
    this->Mass = this->Density * this->FlowVolume;

    if (this->FractionGasLiquid.bIsNotEmpty)
    {
        this->LiquidMass = FractionGasLiquid.LiquidFraction * Mass;
        this->GasMass = FractionGasLiquid.GasFraction * Mass;

        this->LiquidFlowMolar = FractionGasLiquid.LiquidFraction *
FlowMolar;
        this->GasFlowMolar = FractionGasLiquid.GasFraction * FlowMolar;
    }
```

```
}


void UCompound::CalculateVolumeFromMass() {
    this->FlowVolume = this->Mass / this->Density;

}


void UCompound::CalculateFlowMolar() {
    FlowMolar = Mass / WeightMolar;

}

void UCompound::CalculateMass() {
    this->Mass = this->FlowMolar * this->WeightMolar;

    if (this->FractionGasLiquid.bIsNotEmpty)
    {
        this->LiquidFlowMolar = FractionGasLiquid.LiquidFraction * this-
>FlowMolar;
        this->GasFlowMolar = FractionGasLiquid.GasFraction * this-
>FlowMolar;
        this->LiquidMass = this->LiquidFlowMolar * this->WeightMolar;
        this->GasMass = this->GasFlowMolar * this->WeightMolar;
    }
}

void UCompound::PrintProperties() {
    UE_LOG(LogTemp, Warning, TEXT("Name: %s"), *this->Name);
    UE_LOG(LogTemp, Warning, TEXT("Formula: %s"), *this->Formula);
    UE_LOG(LogTemp, Warning, TEXT("Weight Molar: %f"), this->WeightMolar);
    UE_LOG(LogTemp, Warning, TEXT("Concentration in Flow: %f"), this-
>ConcInFlow);
    UE_LOG(LogTemp, Warning, TEXT("Flow Molar: %f"), this->FlowMolar);
    UE_LOG(LogTemp, Warning, TEXT("Mass: %f"), this->Mass);
}




void UMixture::Init(FString inName, FString inStateOfMatter,
TArray<UCompound*> inCompounds)
{
    this->Name = inName;
    this->Compounds = inCompounds;
    this->StateOfMatter = inStateOfMatter;

    if (inStateOfMatter.ToLower().Equals("liquid"))
    {
        this->bIsLiquid = true;
    }
    else if (inStateOfMatter.ToLower().Equals("gas"))
    {
        this->bIsGas = true;
    }
}


void UMixture::CalculateProperties() {
    CalcWeightMolar();
    CalcFlowMolar();
```

```
    CalculateMass();
    SetComposition();
}

void UMixture::CalcWeightMolar() {
    float mols = 0;
    float mass = 0;

    for (auto& compound : this->Compounds) {
        if (bIsGas && compound->FractionGasLiquid.bIsNotEmpty)
        {
            mols += compound->GasMass / compound->WeightMolar;
            mass += compound->GasMass;
            continue;
        }
        else if (bIsLiquid && compound->FractionGasLiquid.bIsNotEmpty)
        {
            mols += compound->LiquidMass / compound->WeightMolar;
            mass += compound->LiquidMass;
            continue;
        }
        mols += compound->Mass / compound->WeightMolar;
        mass += compound->Mass;
    }

    if (mols == 0)
    {
        this->WeightMolar = 0;
        return;
    }
    this->WeightMolar = mass / mols;
}

void UMixture::CalcFlowMolar() {
    float flowMolar = 0;
    for (auto& compound : this->Compounds) {
        if (bIsGas && compound->FractionGasLiquid.bIsNotEmpty)
        {
            flowMolar += compound->GasFlowMolar;
            continue;
        }
        else if (bIsLiquid && compound->FractionGasLiquid.bIsNotEmpty)
        {
            flowMolar += compound->LiquidFlowMolar;
            continue;
        }

        flowMolar += compound->FlowMolar;

    }
    this->FlowMolar = flowMolar;
}

TArray<FCompositionComponent> UMixture::SetCompositionStructure() {
    TArray <FCompositionComponent> CompositionStructure;

    for (auto& compound : this->Compounds)
    {
        CompositionStructure.Add(
            FCompositionComponent(
                compound->Name,
                0,
```

```
                0,
                0));

    }
    return CompositionStructure;
}

void UMixture::SetComposition() {
    double percent;

    if (FlowMolar == 0) {
        this->Composition = SetCompositionStructure();
        return;
    }
    for (auto& compound : this->Compounds)
    {
        if (compound->FractionGasLiquid.bIsNotEmpty)
        {
            if (this->bIsGas) {
                percent = compound->GasMass / this->Mass * 100;
                this->Composition.Add(
                    FCompositionComponent(
                        compound->Name,
                        compound->GasMass,
                        compound->GasFlowMolar,
                        percent
                    )
                );
                continue;
            }
            else if (this->bIsLiquid) {
                percent = compound->LiquidMass / this->Mass * 100;
                this->Composition.Add(
                    FCompositionComponent(
                        compound->Name,
                        compound->LiquidMass,
                        compound->LiquidFlowMolar,
                        percent
                    )
                );
                continue;
            }

        }

        percent = compound->Mass / this->Mass * 100;
        this->Composition.Add(
            FCompositionComponent(
                compound->Name,
                compound->Mass,
                compound->FlowMolar,
                percent
            )
        );

    }

}

void UMixture::PrintProperties() {
    UE_LOG(LogTemp, Warning, TEXT("Name: %s"), *this->Name);
    UE_LOG(LogTemp, Warning, TEXT("Weight Molar: %f"), this->WeightMolar);
```

```
    UE_LOG(LogTemp, Warning, TEXT("Flow Molar: %f"), this->FlowMolar);
    UE_LOG(LogTemp, Warning, TEXT("\nComposition:\n"));
    for (auto& compound : this->Compounds) {
        UE_LOG(LogTemp, Warning, TEXT("   -%s: %f kmol"), *compound->Name,
compound->FlowMolar);
    }
}

UCompound* UMixture::GetCompound(FString inName)
{
    for (auto item : Compounds) {
        FString itemName = item->Name.ToLower();
        inName = inName.ToLower();

        if (itemName ==inName) {
            return item;
        }
    }

    return nullptr;
}

void UMixture::SetConcentration(TArray<FCompositionData> inComposition,
double inFlowIn)
{
    for (auto& item : inComposition) {
        auto compound = GetCompound(item.NameCompound);
        double concInFlow = item.ConcentrationValue;
        compound->ConcInFlow = concInFlow;
        compound->FlowMolar = concInFlow * inFlowIn/ compound->WeightMolar;
    }
    FlowVolume = inFlowIn;

    CalculateProperties();
}
```

APÊNDICE B – CÓDIGO-FONTE PARA CONSTRUÇÃO DE LISTA DE COMPOSTOS

Arquivo Compounds.h

```
#pragma once

#include "CoreMinimal.h"
#include "UObject/NoExportTypes.h"
#include "Compound.h"
#include "Compounds.generated.h"

/**
 *
 */
UCLASS(Blueprintable)
class MYPROJECT2_API UCompounds : public UObject
{
	GENERATED_BODY()

public:
	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"Compounds");
	TArray<UCompound*> Items;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"Compounds");
	TArray<UMixture*> ItemsMixture;

	UCompounds() {
		PopulateList();

	}

	UFUNCTION(BlueprintCallable, Category = "Compounds")
	void PopulateList();

	UFUNCTION(BlueprintCallable, Category = "Compounds")
	UCompound* GetValue(FString Name);

	UFUNCTION(BlueprintCallable, Category = "Compounds")
	UMixture* GetMixture(FString name);

};
```

Arquivo Compounds.cpp

```
#include "Compounds.h"


void UCompounds::PopulateList()
{
    UCompound* aceticAcid = NewObject<UCompound>();
    aceticAcid->Init(1, TEXT("Acetic Acid"), TEXT("C2H4O2"), TEXT("64-19-
7"), TEXT("Conventional"), 60.0526);

    UCompound* water = NewObject<UCompound>();
```

```
    water->Init(2, TEXT("Water"), TEXT("H2O"), TEXT("7732-18-5"),
TEXT("Conventional"), 18.0153);

    UCompound* glucose = NewObject<UCompound>();
    glucose->Init(3, TEXT("Glucose"), TEXT("C6H12O6"), TEXT("50-99-7"),
TEXT("Conventional"), 180.158);

    UCompound* amylose = NewObject<UCompound>();
    amylose->Init(4, TEXT("Amylose"), TEXT("C30H52O26"), TEXT("34620-76-3"),
TEXT("Conventional"), 828.727);

    UCompound* amylopectina = NewObject<UCompound>();
    amylopectina->Init(5, TEXT("Amylopectina"), TEXT("C30H52O26"),
TEXT("34620-76-3"), TEXT("Conventional"), 828.727);

    UCompound* protein = NewObject<UCompound>();
    protein->Init(6, TEXT("Protein"), TEXT("C13H25O7N3S1"), TEXT(""),
TEXT("Conventional"), 367.42);

    UCompound* methane = NewObject<UCompound>();
    methane->Init(7, TEXT("Methane"), TEXT("CH4"), TEXT("74-82-8"),
TEXT("Conventional"), 16.0428);

    UCompound* carbonDioxide = NewObject<UCompound>();
    carbonDioxide->Init(8, TEXT("Carbon Dioxide"), TEXT("CO2"), TEXT("124-
38-9"), TEXT("Conventional"), 44.0098);

    UCompound* ammonia = NewObject<UCompound>();
    ammonia->Init(9, TEXT("Ammonia"), TEXT("NH3"), TEXT("7664-41-7"),
TEXT("Conventional"), 17.0306);

    UCompound* hydrogenSulfide = NewObject<UCompound>();
    hydrogenSulfide->Init(10, TEXT("Hydrogen Sulfide"), TEXT("H2S"),
TEXT("7783-06-3"), TEXT("Conventional"), 34.0819);

    UCompound* triolein = NewObject<UCompound>();
    triolein->Init(11, TEXT("Triolein"), TEXT("C57H104O6"), TEXT("122-32-
7"), TEXT("Conventional"), 885.449);

    UCompound* glycerol = NewObject<UCompound>();
    glycerol->Init(12, TEXT("Glycerol"), TEXT("C3H8O3"), TEXT("56-81-5"),
TEXT("Conventional"), 92.0947);

    UCompound* oleicAcid = NewObject<UCompound>();
    oleicAcid->Init(13, TEXT("Oleic Acid"), TEXT("C18H34O2"), TEXT("112-80-
1"), TEXT("Conventional"), 282.467);

    UCompound* microorganism = NewObject<UCompound>();
    microorganism->Init(14, TEXT("Microorganism"), TEXT("C5H7O2N1"),
TEXT(""), TEXT("Conventional"), 113.116);

    UCompound* propionicAcid = NewObject<UCompound>();
    propionicAcid->Init(15, TEXT("Propionic Acid"), TEXT("C3H6O2"),
TEXT("79-09-4"), TEXT("Conventional"), 74.0794);

    UCompound* butyricAcid = NewObject<UCompound>();
    butyricAcid->Init(16, TEXT("Butyric Acid"), TEXT("C4H8O2"), TEXT("107-
92-6"), TEXT("Conventional"), 88.1063);

    UCompound* lacticAcid = NewObject<UCompound>();
    lacticAcid->Init(17, TEXT("Lactic Acid"), TEXT("C3H6O3"), TEXT("50-21-
5"), TEXT("Conventional"), 90.0788);
```

```
    UCompound* hydrogen = NewObject<UCompound>();
    hydrogen->Init(18, TEXT("Hydrogen"), TEXT("H2"), TEXT("1333-74-0"),
TEXT("Conventional"), 2.01588);

    UCompound* oxygen = NewObject<UCompound>();
    oxygen->Init(19, TEXT("Oxygen"), TEXT("O2"), TEXT("7782-44-7"),
TEXT("Conventional"), 15.999);

    UCompound* nitrogen = NewObject<UCompound>();
    nitrogen->Init(20, TEXT("Nitrogen"), TEXT("N2"), TEXT("7727-37-9"),
TEXT("Conventional"), 28.02);

    UCompound* ethanol = NewObject<UCompound>();
    ethanol->Init(21, TEXT("Ethanol"), TEXT("C2H6O2"), TEXT("64-17-5"),
TEXT("Conventional"), 46.069);

    TArray<UCompound*> ListCompounds = { ethanol, aceticAcid, water,
glucose, amylose, amylopectina, protein, methane, carbonDioxide,
        ammonia, hydrogenSulfide, triolein, glycerol, oleicAcid,
microorganism, propionicAcid, butyricAcid, lacticAcid, hydrogen,
        oxygen, nitrogen , ethanol};

    Items.Append(ListCompounds);

    UMixture* Biomass = NewObject<UMixture>();
    TArray<UCompound*> BiomassCompounds = { amylopectina, amylose, water,
ethanol, aceticAcid, protein, ammonia, triolein, propionicAcid, butyricAcid,
lacticAcid };

    Biomass->Init("Biomass", "Liquid", BiomassCompounds);
    ItemsMixture.Add(Biomass);
}

UCompound* UCompounds::GetValue(FString name)
{
    for (auto item : Items) {
        FString itemName = item->Name.ToLower();
        name = name.ToLower();

        if (itemName == name) {
            return item;
        }
    }

    return nullptr;
}


UMixture* UCompounds::GetMixture(FString name)
{
    for (auto item : ItemsMixture) {
        FString itemName = item->Name.ToLower();
        name = name.ToLower();

        if (itemName == name) {
            return item;
        }
    }

    return nullptr;
}
```

APÊNDICE C – CÓDIGO-FONTE PARA CRIAÇÃO DE REAÇÃO

Arquivo Reaction.h

```
#pragma once

#include "Containers/Map.h"
#include "Compound.h"
#include "ReactionSetting.h"
#include "UObject/NoExportTypes.h"
#include "CoreMinimal.h"
#include "Reaction.generated.h"


UCLASS(Blueprintable)
class MYPROJECT2_API UReaction : public UObject
{
	GENERATED_BODY()

public:
    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    int32 OrderReaction;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    float ConversionFactor;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    FString StepReaction;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    bool IsReverse;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    TArray<FReactionComponent> Reagents;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    TArray<FReactionComponent> Products;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    double FlowInlet = 0;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    double ReactionExtent;

    UReaction(){}

    void Init(int32 inOrderReaction, float inConversionFactor, FString
inStepReaction, bool inIsReverse);

    UFUNCTION(BlueprintCallable, Category = "Reactor")
    void Start();

    UFUNCTION(BlueprintCallable, Category = "Reactor")
    void CalcFlowMol();

    UFUNCTION(BlueprintCallable, Category = "Reactor")
    void CalcReactionExtent();

    UFUNCTION(BlueprintCallable, Category = "Reactor")
```

```
    void CalcFlowOutlet();

    UFUNCTION(BlueprintCallable, Category = "Reactor")
    void AddReagentProduct(FReactionSetting reactionSet);

};
```

Arquivo Reaction.cpp

```
#include "Reaction.h"

void UReaction::Init(int32 inOrderReaction, float inConversionFactor,
FString inStepReaction, bool inIsReverse) {
    this->OrderReaction = inOrderReaction;
    this->ConversionFactor = inConversionFactor;
    this->StepReaction = inStepReaction;
    this->IsReverse = inIsReverse;
}

void UReaction::Start()
{
    CalcFlowMol();
    CalcReactionExtent();
    CalcFlowOutlet();


}

void UReaction::CalcFlowMol()
{
    for (auto& reagent : Reagents) {
        if (!reagent.Compound->FlowMolar) {
            reagent.Compound->FlowMolar += reagent.Compound->ConcInFlow *
this->FlowInlet / reagent.Compound->WeightMolar;
        }
    }

    for (auto& product : Products) {
        if (!product.Compound->FlowMolar) {
            product.Compound->FlowMolar += product.Compound->ConcInFlow *
this->FlowInlet / product.Compound->WeightMolar;
        }
    }
}

void UReaction::CalcReactionExtent()
{
    double reactionExtentValue;
    double reactionExtentValueMin = 10;
    bool isFirst = true;

    for (auto& reagent : Reagents) {
        if (isFirst)
        {
            reactionExtentValueMin = -reagent.Compound->FlowMolar /
reagent.Coef;
            isFirst = false;
            continue;
        }

        reactionExtentValue = -reagent.Compound->FlowMolar / reagent.Coef;
```

```
        if (reactionExtentValue < reactionExtentValueMin)
        {
            reactionExtentValueMin = reactionExtentValue;
        }

    }

    this->ReactionExtent = reactionExtentValueMin * this->ConversionFactor;
}

void UReaction::CalcFlowOutlet()
{
    for (auto& reagent : this->Reagents) {
        reagent.Compound->FlowMolar = reagent.Compound->FlowMolar +
reagent.Coef * this->ReactionExtent;
        reagent.Compound->CalculateMass();
    }

    for (auto& product : this->Products) {
        product.Compound->FlowMolar = product.Compound->FlowMolar +
product.Coef * this->ReactionExtent;
        product.Compound->CalculateMass();
    }
}

void UReaction::AddReagentProduct(FReactionSetting reactionSet)
{
    this->Reagents.Append(reactionSet.Reagents);
    this->Products.Append(reactionSet.Products);
}
```

APÊNDICE D – CÓDIGO-FONTE PARA CONSTRUÇÃO DE LISTA DE REAÇÕES

Arquivo Reactions.h

```
#pragma once

#include "CoreMinimal.h"
#include "UObject/NoExportTypes.h"
#include "Reaction.h"
#include "Reactions.generated.h"

UCLASS(Blueprintable)
class MYPROJECT2_API UReactions : public UObject
{
        GENERATED_BODY()
public:
    TArray<UReaction*> Items;

    UReactions() {
        PopulateList();
    }

    UFUNCTION(BlueprintCallable, Category = "Reactor")
    void PopulateList();

};
```

Arquivo Reactions.cpp

```
#include "Reactions.h"

void UReactions::PopulateList() {
    UReaction* reaction1 = NewObject<UReaction>();
    reaction1->Init(1, 1.0, TEXT("Hydrolisis"), false);

    UReaction* reaction2 = NewObject<UReaction>();
    reaction2->Init(2, 1.0, TEXT("Hydrolisis"), false);

    UReaction* reaction3 = NewObject<UReaction>();
    reaction3->Init(3, 0.538561856111995, TEXT("Hydrolisis"), false);

    UReaction* reaction4 = NewObject<UReaction>();
    reaction4->Init(4, 1, TEXT("Hydrolisis"), false);

    UReaction* reaction5 = NewObject<UReaction>();
    reaction5->Init(5, 0.33, TEXT("Acidogenesis"), false);

    UReaction* reaction6 = NewObject<UReaction>();
    reaction6->Init(6, 0.50, TEXT("Acidogenesis"), false);

    UReaction* reaction7 = NewObject<UReaction>();
    reaction7->Init(7, 1, TEXT("Hydrolisis"), false);

    UReaction* reaction8 = NewObject<UReaction>();
    reaction8->Init(8, 1, TEXT("Acidogenesis"), false);

    UReaction* reaction9 = NewObject<UReaction>();
    reaction9->Init(9, 1, TEXT("Acetogenesis"), false);

    UReaction* reaction10 = NewObject<UReaction>();
    reaction10->Init(10, 1, TEXT("Acetogenesis"), false);

    UReaction* reaction11 = NewObject<UReaction>();
    reaction11->Init(11, 1, TEXT("Acidogenesis"), false);

    UReaction* reaction12 = NewObject<UReaction>();
    reaction12->Init(12, 1, TEXT("Acetogenesis"), false);

    UReaction* reaction13 = NewObject<UReaction>();
    reaction13->Init(13, 1, TEXT("Hydrolisis"), false);

    UReaction* reaction14 = NewObject<UReaction>();
    reaction14->Init(14, 1, TEXT("Acetotrophic Methanogenesis"), false);

    UReaction* reaction15 = NewObject<UReaction>();
    reaction15->Init(15, 1, TEXT("Hydrogenotrophic Methanogenesis"), false);

    Items = { reaction1, reaction2, reaction3, reaction4, reaction5,
reaction6, reaction7, reaction8, reaction9, reaction10, reaction11,
reaction12, reaction13, reaction14, reaction15 };
}
```

APÊNDICE E – CÓDIGO-FONTE PARA LEITURA DE COMPOSTOS E REAÇÕES E CÁLCULOS DO BIODIGESTOR

Arquivo Reactor.h

```
#pragma once

#include "CoreMinimal.h"
#include "Kismet/KismetStringLibrary.h"
#include "GameFramework/Actor.h"
#include "UObject/NoExportTypes.h"
#include "Containers/Map.h"
#include "Reaction.h"
#include "ReactionSetting.h"
#include "Composition.h"
#include "Compound.h"
#include "Reactions.h"
#include "Compounds.h"
#include "Reactor.generated.h"


UCLASS(Blueprintable)
class MYPROJECT2_API UReactor : public UObject
{
	GENERATED_BODY()

public:
	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
	double Flow;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
	TArray<FCompositionData> Composition;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
	UCompounds* Compounds = NewObject<UCompounds>();

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
	UMixture* Biogas = NewObject<UMixture>();

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
	UMixture* Digestate = NewObject<UMixture>();

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
	UReactions* Reactions = NewObject<UReactions>();

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
	TArray<UCompound*> BiogasCompounds;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
	TArray<UCompound*> DigestateCompounds;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
	TArray<FReactionSetting> ReactionSetting;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
	TArray<UCompound*> GasesToDissolve;
```

```
        UReactor(){}

        UFUNCTION(BlueprintCallable, Category = "Reactor")
        void StartReactor(const double inFlow, const
TArray<FCompositionData> inComposition);

        UFUNCTION(BlueprintCallable, Category = "Reactor")
        void SetConcentration();

        UFUNCTION(BlueprintCallable, Category = "Reactor")
        void DissolveGases(TArray<UCompound*> inGasesToDissolve);

        UFUNCTION(BlueprintCallable, Category = "Reactor")
        int GetCountElement(FString inFormula, FString inElement);

        UFUNCTION(BlueprintCallable, Category = "Reactor")
        void LoadReactions();


};
```

Arquivo Reactor.cpp

```
#include "Reactor.h"

void UReactor::StartReactor(const double inFlow, const
TArray<FCompositionData> inComposition)
{
    this->Flow = inFlow;
    this->Composition = inComposition;

    SetConcentration();
    LoadReactions();

    for (auto& reaction : Reactions->Items) {
        reaction->Start();
    }
    DissolveGases(GasesToDissolve);

    Biogas->Init("Biogas", "gas", BiogasCompounds);
    Digestate->Init("Digestate", "liquid", DigestateCompounds);

    this->Biogas->CalculateProperties();
    this->Digestate->CalculateProperties();

    this->Biogas->PrintProperties();
    this->Digestate->PrintProperties();
}

void UReactor::SetConcentration()
{
    for (auto& item : Composition) {
        auto compound = Compounds->GetValue(item.NameCompound);
        double concInFlow = item.ConcentrationValue;
        compound->ConcInFlow = concInFlow;
    }
}

int UReactor::GetCountElement(FString inFormula, FString inElement)
{
    int countElement;
```

```
    TArray<FString> inFormulaArr =
UKismetStringLibrary::GetCharacterArrayFromString(inFormula);

    for (int index = 0; index < inFormulaArr.Num(); index++)
    {
        FString currentChar = inFormulaArr[index];

        if (currentChar.Equals(inElement))
        {
            if (inFormulaArr.IsValidIndex(index + 1))
            {
                FString nextChar = inFormulaArr[index + 1];
                if (nextChar.IsNumeric())
                {
                    countElement = FCString::Atoi(*nextChar);
                    return countElement;
                }
            }
            countElement = 1;
            return countElement;
        }
    }
    return 0;
}

void UReactor::DissolveGases(TArray<UCompound*> inGasesToDissolve)
{
    UCompound* Helementary = NewObject<UCompound>();
    UCompound* Celementary = NewObject<UCompound>();
    UCompound* Nelementary = NewObject<UCompound>();
    UCompound* Selementary = NewObject<UCompound>();
    UCompound* Oelementary = NewObject<UCompound>();

    TArray<UCompound*> ElementariesArr = { Helementary, Celementary,
Nelementary, Selementary, Oelementary };

    TArray<TTuple<FString, double>> Elementaries = {
    {"H", 1.00784, },
    {"C", 12.011, },
    {"N", 14.0067, },
    {"S", 32.065, },
    {"O", 15.999, }
    };


    double flowMolarElement;
    int id = 22;
    int index = 0;

    for (auto& element : Elementaries) {
        flowMolarElement = 0;
        FString SymbolElement = element.Get<0>();
        double WeightMolarElement = element.Get<1>();

        for (auto& Gas : inGasesToDissolve) {
            FString formula = Gas->Formula;
            if (Gas->Formula.Contains(SymbolElement))
            {
                int CountElement = GetCountElement(formula, SymbolElement);
                flowMolarElement += WeightMolarElement * CountElement / Gas-
>WeightMolar * Gas->LiquidFlowMolar;
            }
```

```
        }

        UCompound* ElementaryObject = ElementariesArr[index];
        ElementaryObject->Init(id, SymbolElement, SymbolElement, TEXT(""),
TEXT("Conventional"), WeightMolarElement);
        ElementaryObject->FlowMolar = flowMolarElement;

        id++;
        index++;
    }

    for (auto& elementar : ElementariesArr)
    {

        elementar->CalculateMass();
        DigestateCompounds.Add(elementar);

    }
}


void UReactor::LoadReactions()
{
    auto water = Compounds->GetValue("water");
    auto glucose = Compounds->GetValue("glucose");
    auto amylopectina = Compounds->GetValue("amylopectina");
    auto amylose = Compounds->GetValue("amylose");
    auto protein = Compounds->GetValue("protein");
    auto methane = Compounds->GetValue("methane");
    auto carbon_dioxide = Compounds->GetValue("carbon dioxide");
    auto ammonia = Compounds->GetValue("ammonia");
    auto hydrogen_sulfide = Compounds->GetValue("hydrogen sulfide");
    auto triolein = Compounds->GetValue("triolein");
    auto glycerol = Compounds->GetValue("glycerol");
    auto oleic_acid = Compounds->GetValue("oleic acid");
    auto microorganism = Compounds->GetValue("microorganism");
    auto acetic_acid = Compounds->GetValue("acetic acid");
    auto propionic_acid = Compounds->GetValue("propionic acid");
    auto butyric_acid = Compounds->GetValue("butyric acid");
    auto lactic_acid = Compounds->GetValue("lactic acid");
    auto hydrogen = Compounds->GetValue("hydrogen");
    auto ethanol = Compounds->GetValue("ethanol");

    double fractionLiquidWater = 0.997836428066458;
    water->FractionGasLiquid = FGasLiquid(fractionLiquidWater);

    double fractionLiquidCarbonDioxide = 0.0246629061518668;
    carbon_dioxide->FractionGasLiquid =
FGasLiquid(fractionLiquidCarbonDioxide);

    double fractionLiquidMethane = 0.00132064749238705;
    methane->FractionGasLiquid = FGasLiquid(fractionLiquidMethane);

    double fractionLiquidAmmonia = 0.961480179212209;
    ammonia->FractionGasLiquid = FGasLiquid(fractionLiquidAmmonia);

    double fractionLiquidHydrogenSulfide = 0.0757422936709186;
    hydrogen_sulfide->FractionGasLiquid =
FGasLiquid(fractionLiquidHydrogenSulfide);

    BiogasCompounds = { carbon_dioxide, methane, ammonia, hydrogen_sulfide,
hydrogen, water };
```

```
    GasesToDissolve = { methane, ammonia, hydrogen_sulfide, carbon_dioxide
};
    DigestateCompounds = {water, protein};

    TArray<FReactionComponent> ReagentsReaction1;
    TArray<FReactionComponent> ProductsReaction1;
    ReagentsReaction1 = TArray<FReactionComponent>{FReactionComponent(-4,
water), FReactionComponent(-1, amylopectina)};
    ProductsReaction1 = TArray<FReactionComponent>{FReactionComponent(5,
glucose)};

    TArray<FReactionComponent> ReagentsReaction2;
    TArray<FReactionComponent> ProductsReaction2;
    ReagentsReaction2 = TArray<FReactionComponent>{FReactionComponent(-4,
water), FReactionComponent(-1, amylose)};
    ProductsReaction2 = TArray<FReactionComponent>{FReactionComponent(5,
glucose)};

    TArray<FReactionComponent> ReagentsReaction3;
    TArray<FReactionComponent> ProductsReaction3;
    ReagentsReaction3 = TArray<FReactionComponent>{FReactionComponent(-1,
protein), FReactionComponent(-6, water)};
    ProductsReaction3 = TArray<FReactionComponent>{FReactionComponent(6.5,
methane), FReactionComponent(6.5, carbon_dioxide), FReactionComponent(3,
ammonia), FReactionComponent(1, hydrogen_sulfide)};

    TArray<FReactionComponent> ReagentsReaction4;
    TArray<FReactionComponent> ProductsReaction4;
    ReagentsReaction4 = TArray<FReactionComponent>{FReactionComponent(-3,
water), FReactionComponent(-1, triolein)};
    ProductsReaction4 = TArray<FReactionComponent>{FReactionComponent(1,
glycerol), FReactionComponent(3, oleic_acid)};

    TArray<FReactionComponent> ReagentsReaction5;
    TArray<FReactionComponent> ProductsReaction5;
    ReagentsReaction5 = TArray<FReactionComponent>{FReactionComponent(-1,
glucose), FReactionComponent(-0.1115, ammonia)};
    ProductsReaction5 =
TArray<FReactionComponent>{FReactionComponent(0.1115, microorganism),
FReactionComponent(0.744, acetic_acid), FReactionComponent(0.5,
propionic_acid), FReactionComponent(0.4409, butyric_acid),
FReactionComponent(0.6909, carbon_dioxide), FReactionComponent(1.0254,
water)};

    TArray<FReactionComponent> ReagentsReaction6;
    TArray<FReactionComponent> ProductsReaction6;
    ReagentsReaction6 = TArray<FReactionComponent>{FReactionComponent(-1,
glucose)};
    ProductsReaction6 = TArray<FReactionComponent>{FReactionComponent(2,
lactic_acid)};

    TArray<FReactionComponent> ReagentsReaction7;
    TArray<FReactionComponent> ProductsReaction7;
    ReagentsReaction7 = TArray<FReactionComponent>{FReactionComponent(-1,
glucose)};
    ProductsReaction7 = TArray<FReactionComponent>{FReactionComponent(2,
ethanol), FReactionComponent(2, carbon_dioxide)};

    TArray<FReactionComponent> ReagentsReaction8;
    TArray<FReactionComponent> ProductsReaction8;
    ReagentsReaction8 = TArray<FReactionComponent>{FReactionComponent(-3,
lactic_acid)};
```

```
    ProductsReaction8 = TArray<FReactionComponent>{FReactionComponent(2,
propionic_acid), FReactionComponent(1, acetic_acid), FReactionComponent(1,
carbon_dioxide), FReactionComponent(1, water)};

    TArray<FReactionComponent> ReagentsReaction9;
    TArray<FReactionComponent> ProductsReaction9;
    ReagentsReaction9 = TArray<FReactionComponent>{FReactionComponent(-
15.2396, water), FReactionComponent(-1, oleic_acid), FReactionComponent(-
0.1701, ammonia), FReactionComponent(-0.2501, carbon_dioxide)};
    ProductsReaction9 =
TArray<FReactionComponent>{FReactionComponent(0.1701, microorganism),
FReactionComponent(8.6998, acetic_acid), FReactionComponent(14.4978,
hydrogen)};

    TArray<FReactionComponent> ReagentsReaction10;
    TArray<FReactionComponent> ProductsReaction10;
    ReagentsReaction10 = TArray<FReactionComponent>{FReactionComponent(-
0.8038, water), FReactionComponent(-0.0653, ammonia), FReactionComponent(-1,
butyric_acid), FReactionComponent(-0.0006, hydrogen), FReactionComponent(-
0.5543, carbon_dioxide)};
    ProductsReaction10 =
TArray<FReactionComponent>{FReactionComponent(0.0653, microorganism),
FReactionComponent(1.8909, acetic_acid), FReactionComponent(0.446,
methane)};

    TArray<FReactionComponent> ReagentsReaction11;
    TArray<FReactionComponent> ProductsReaction11;
    ReagentsReaction11 = TArray<FReactionComponent>{FReactionComponent(-1,
glycerol), FReactionComponent(-0.04071, ammonia), FReactionComponent(-
0.0291, carbon_dioxide), FReactionComponent(-0.0005, hydrogen)};
    ProductsReaction11 =
TArray<FReactionComponent>{FReactionComponent(0.04071, microorganism),
FReactionComponent(0.94185, propionic_acid), FReactionComponent(1.09308,
water)};

    TArray<FReactionComponent> ReagentsReaction12;
    TArray<FReactionComponent> ProductsReaction12;
    ReagentsReaction12 = TArray<FReactionComponent>{FReactionComponent(-
0.31434, water), FReactionComponent(-1, propionic_acid),
FReactionComponent(-0.06198, ammonia)};
    ProductsReaction12 =
TArray<FReactionComponent>{FReactionComponent(0.06198, microorganism),
FReactionComponent(0.9345, acetic_acid), FReactionComponent(0.66041,
methane), FReactionComponent(0.16069, carbon_dioxide),
FReactionComponent(0.00055, hydrogen)};

    TArray<FReactionComponent> ReagentsReaction13;
    TArray<FReactionComponent> ProductsReaction13;
    ReagentsReaction13 = TArray<FReactionComponent>{FReactionComponent(-2,
ethanol), FReactionComponent(-1, carbon_dioxide)};
    ProductsReaction13 = TArray<FReactionComponent>{FReactionComponent(2,
acetic_acid), FReactionComponent(1, methane)};

    TArray<FReactionComponent> ReagentsReaction14;
    TArray<FReactionComponent> ProductsReaction14;
    ReagentsReaction14 = TArray<FReactionComponent>{FReactionComponent(-
0.022, ammonia), FReactionComponent(-1, acetic_acid)};
    ProductsReaction14 =
TArray<FReactionComponent>{FReactionComponent(0.022, microorganism),
FReactionComponent(0.945, methane), FReactionComponent(0.945,
carbon_dioxide), FReactionComponent(0.066, water)};
```

```
    TArray<FReactionComponent> ReagentsReaction15;
    TArray<FReactionComponent> ProductsReaction15;
    ReagentsReaction15 = TArray<FReactionComponent>{FReactionComponent(-
14.4976, hydrogen), FReactionComponent(-3.8334, carbon_dioxide),
FReactionComponent(-0.0836, ammonia)};
    ProductsReaction15 =
TArray<FReactionComponent>{FReactionComponent(0.0836, microorganism),
FReactionComponent(3.4154, methane), FReactionComponent(7.4996, water)};

    ReactionSetting = TArray<FReactionSetting>{
        FReactionSetting(1, ReagentsReaction1, ProductsReaction1),
        FReactionSetting(2, ReagentsReaction2, ProductsReaction2),
        FReactionSetting(3, ReagentsReaction3, ProductsReaction3),
        FReactionSetting(4, ReagentsReaction4, ProductsReaction4),
        FReactionSetting(5, ReagentsReaction5, ProductsReaction5),
        FReactionSetting(6, ReagentsReaction6, ProductsReaction6),
        FReactionSetting(7, ReagentsReaction7, ProductsReaction7),
        FReactionSetting(8, ReagentsReaction8, ProductsReaction8),
        FReactionSetting(9, ReagentsReaction9, ProductsReaction9),
        FReactionSetting(10, ReagentsReaction10, ProductsReaction10),
        FReactionSetting(11, ReagentsReaction11, ProductsReaction11),
        FReactionSetting(12, ReagentsReaction12, ProductsReaction12),
        FReactionSetting(13, ReagentsReaction13, ProductsReaction13),
        FReactionSetting(14, ReagentsReaction14, ProductsReaction14),
        FReactionSetting(15, ReagentsReaction15, ProductsReaction15)
    };


    for (auto& reaction : Reactions->Items) {
        reaction->FlowInlet = this->Flow;
        int orderReaction = reaction->OrderReaction;

        for (auto& reactionSet : ReactionSetting) {
            if (reactionSet.Id == orderReaction) {
                reaction->AddReagentProduct(reactionSet);
            }
        }
    }
}
```

APÊNDICE F – CÓDIGO-FONTE PARA CÁLCULOS DO TROCADOR DE CALOR

Arquivo ShellTube.h

```
#pragma once

#include "CoreMinimal.h"
#include "UObject/NoExportTypes.h"
#include "Compound.h"
#include "ShellTube.generated.h"

/**
 *
 */
UCLASS(Blueprintable)
class MYPROJECT2_API UShellTube : public UObject
{
	GENERATED_BODY()

public:

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"ShellTube")
	double FlowTube;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"ShellTube")
	double FlowShell;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"ShellTube")
	double Heat;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"ShellTube")
	double CPTube;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"ShellTube")
	double CPShell;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"ShellTube")
	double TemperatureTubeIn;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"ShellTube")
	double TemperatureTubeOut;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"ShellTube")
	double TemperatureShellIn;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"ShellTube")
	double TemperatureShellOut;

	UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"ShellTube")
```

```
        UCompound* ShellCompound;

        UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category =
"ShellTube")
        UCompound* TubeCompound;


        UShellTube();

        UFUNCTION(BlueprintCallable, Category = "ShellTube")
        void StartHeater(const double inTShellOut, const double inTTubeOut);

        UFUNCTION(BlueprintCallable, Category = "ShellTube")
        void LoadTubeIn(UCompound* inCompound);

        UFUNCTION(BlueprintCallable, Category = "ShellTube")
        void LoadShellIn(UCompound* inCompound);

        void LoadTubeIn(UMixture* inCompound);
        void LoadShellIn(UMixture* inCompound);

        UFUNCTION(BlueprintCallable, Category = "ShellTube")
        void EnergyBalance();

        UFUNCTION(BlueprintCallable, Category = "ShellTube")
        void ConvertFromCalToKJ();
};
```

Arquivo ShellTube.cpp

```
#include "ShellTube.h"

UShellTube::UShellTube() {}

void UShellTube::StartHeater(const double inTShellOut, const double
inTTubeOut)
{
        TemperatureShellOut = inTShellOut;
        TemperatureTubeOut = inTTubeOut;

        EnergyBalance();
}

void UShellTube::LoadTubeIn(UCompound* inCompound) {
        TubeCompound = inCompound;
        TemperatureTubeIn = inCompound->Temperature;
        CPTube = inCompound->HeatCapacity;
        FlowTube = inCompound->FlowMolar;
}

void UShellTube::LoadShellIn(UCompound* inCompound) {
        ShellCompound = inCompound;
        TemperatureShellIn = inCompound->Temperature;
        CPShell = inCompound->HeatCapacity;
        FlowShell = inCompound->FlowMolar;
}


void UShellTube::LoadTubeIn(UMixture* inCompound) {
        TubeCompound = inCompound;
        TemperatureTubeIn = inCompound->Temperature;
```

```
        FlowTube = inCompound->FlowMolar;
}

void UShellTube::LoadShellIn(UMixture* inCompound) {
        ShellCompound = inCompound;
        TemperatureShellIn = inCompound->Temperature;
        FlowShell = inCompound->FlowMolar;
}

void UShellTube::ConvertFromCalToKJ() {
        Heat = Heat * 0.004184;
}

void UShellTube::EnergyBalance()
{
        Heat = FlowShell * CPShell * (TemperatureShellOut -
TemperatureShellIn);
        FlowTube = abs(Heat / (CPTube * (TemperatureTubeOut -
TemperatureTubeIn)));
        TubeCompound->FlowMolar = FlowTube;
        TubeCompound->Temperature = TemperatureTubeOut;
        ShellCompound->Temperature = TemperatureShellOut;
        TubeCompound->CalculateMass();
        TubeCompound->CalculateVolumeFromMass();
        FlowTube = TubeCompound->FlowVolume;
        FlowShell = ShellCompound->FlowVolume;


        ConvertFromCalToKJ();
}
```

APÊNDICE G – CÓDIGO-FONTE PARA CONSTRUÇÃO DE ESTRUTURAS UTILIZADAS EM OUTROS CÓDIGOS

Arquivo Composition.h

```
#pragma once

#include "CoreMinimal.h"
#include "Containers/Map.h"
#include "Engine/DataTable.h"
#include "Composition.generated.h"

USTRUCT(BlueprintType)
struct FCompositionData : public FTableRowBase
{
    GENERATED_USTRUCT_BODY()

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Composition")
    FString NameCompound;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Composition")
    double ConcentrationValue;

    FCompositionData() {}
};
```

Arquivo ReactionSetting.h

```
#pragma once

#include "CoreMinimal.h"
#include "Containers/Map.h"
#include "Compound.h"
#include "Engine/DataTable.h"
#include "ReactionSetting.generated.h"


USTRUCT(BlueprintType)
struct FReactionComponent
{
    GENERATED_BODY()

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    double Coef;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    UCompound* Compound;

    FReactionComponent(){}

    FReactionComponent(const double InCoef, UCompound* InCompound) :
Coef(InCoef), Compound(InCompound) {}

};

USTRUCT(BlueprintType)
struct FReactionSetting
{
    GENERATED_BODY()

    int32 Id;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    TArray<FReactionComponent> Reagents;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Reactor")
    TArray<FReactionComponent> Products;

    FReactionSetting() {}

    FReactionSetting(const int32 InId, const TArray<FReactionComponent>
InReagents, const TArray<FReactionComponent> InProducts)
        : Id(InId), Reagents(InReagents), Products(InProducts) {}
};
```