Mestrado em Engenharia Informática
Dissertação
Relatório Final

Uso de Criatividade na Construção de Modelos de Legos

João Parreira

joaofp@student.dei.uc.pt

Orientador:

Professor Doutor Pedro Abreu

Co-orientador:

Professor Doutor Daniel Silva

Data: 03 de Julho de 2013

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Resumo

Muitas vezes se tentou encontrar uma definição única para a Criatividade, mas ainda não foi possível, pois é um fenómeno natural que é observado nos seres humanos que ainda não está totalmente compreendido. Em conjunto com a área dos computadores, existem várias tentativas de se fazer programas criativos, ou seja, juntar criatividade e máquinas. Alguns foram melhor sucedidos que outros e existem alguns sucessos, de qualquer das formas em nenhum deles se conseguiu atingir o mais alto nível de criatividade máquina.

Neste trabalho pretende-se criar um sistema que, no seu conjunto seja criativo, conseguindo criar modelos Lego apelativos de forma autónoma e eficaz. Através de um algoritmo genético que vai evoluir a população de Modelos Lego no sistema, em conjunto com o Google Imagens e um motor de física, é calculada um valor de aptidão final para cada modelo. Através desse valor consegue-se aferir se o Modelo é ou não considerado criativo pelo sistema.

Foram efectuados dois conjuntos de testes, sendo o primeiro para verificar qual o melhor valor de cruzamento e mutação a usar na segunda fase, não tendo sido usado o Google Imagens nesta fase. No segundo conjunto, testou-se o funcionamento completo do sistema, usando os valores obtidos anteriormente. Os resultados de uma forma geral demonstram que o sistema pode ser considerado criativo, produzindo na maior parte das vezes algo que não é apelativo à vista humana, mas enquadrando-se em algumas das definições de criatividade. Para trabalho futuro existem muitas possibilidades a seguir, podendo o sistema ser melhorado e apresentar resultados mais aliciantes.

Palavras-Chave: Algoritmo genético; Axiom 3D Rendering Engine; BulletSharp Physics Library; Criatividade; Google Imagens; Lego; Modelos Lego.

Agradecimentos

Gostaria de agradecer a algumas pessoas, sem as quais este trabalho de mestrado não tinha sido possível, que me ajudaram a crescer e a ser o que sou hoje.

Ao orientador desta tese, o Professor Doutor Pedro Abreu, que acreditou desde o início que seria possível efectuar um bom trabalho, conseguindo moldar as várias ideias que me surgiram no enquadramento do tema. Mesmo tendo por vezes o tempo muito à justa, conseguiu sempre analisar as minhas dúvidas, detectar situações menos correctas e dar a sua opinião sábia em tempo útil. Ao co-orientador Professor Doutor Daniel Silva, que esteve presente na maior parte do projecto, em especial na fase inicial, onde se tomaram todas as grandes decisões do projecto.

A todos os professores que me acompanharam ao longo de todo o meu percurso académico, que regra geral, sempre me tentaram motivar e puxar por mim, conseguindo que me superasse, atingindo objectivos ainda mais elevados.

Aos meus colegas de trabalho da PTSI de Coimbra, pela compreensão desta minha etapa académica e óptimo ambiente de trabalho proporcionado.

Aos meus pais, que apesar de não poder estar perto deles, sempre me apoiaram em todas as minhas decisões e me deram a força, educação e a vontade de crescer que vai dentro de mim. Em especial à minha mãe, que sempre me incentivou na construção do meu futuro e na perseguição dos meus sonhos.

À minha namorada, Patrícia Marques, que foi a principal "culpada" por me ter inscrito neste mestrado, um objectivo meu, mas que sem o incentivo dela possivelmente não o teria feito tão rapidamente. Pela ajuda incansável, que mesmo não percebendo muito da área, ouvia as minhas dúvidas e conseguia dar bons conselhos. Por todo o apoio, incentivo e paciência dispendida a rever tudo ao mais ínfimo detalhe, e principalmente, por ser como é.

A todos os meus amigos, que por vezes mesmo sem saberem, me fazem ter a vontade e a motivação necessária para seguir em frente em todos os meus desafios.

Um muito obrigado a todos,

João Daniel Dourado Fernandes Parreira

Índice

Lista de Figuras

Lista de Tabelas

Lista de Equações

1. Introdução

Neste capítulo é apresentado o contexto e motivação do projecto que levou ao seu desenvolvimento e quais os seus principais objectivos. No capítulo 1.1 descreve-se o contexto e motivação. O capítulo 1.2 contém um resumo dos principais objectivos seguido de um capítulo sobre o planeamento efectuado do projecto. Por fim, no capítulo 1.4, resume-se a estrutura e organização deste relatório.

1.1. Contexto e Motivação

Considerada como uma das características mais impressionantes do ser humano, a criatividade é uma área relativamente recente de investigação. O seu estudo envolve uma série de áreas, onde ainda existe muito espaço a descobertas. É um fenómeno natural que é observado nos seres humanos e como tal merece uma explicação rigorosa que, para poder ser ensinada deve ser primeiro compreendida [1]. A forma como conhecemos a criatividade, leva-nos a não conseguir definir bem o seu significado, muito se tem feito para o tentar fazer, mas o que existe são apenas definições muito abstractas.

Uma ideia pode ser considerada criativa por algumas pessoas enquanto por outras pode ser considerada apenas como nova. Na sua classificação de criatividade não chega uma resposta do tipo binário, temos antes de perguntar "quão criativa é e de que modo" e para se poder considerar a ideia criativa, além de nova, tem de ser surpreendente e valiosa [2]. Vamos focar-nos numa subárea da criatividade que também é abordada, para além do tema da mente criativa e alguns dos seus mitos e mecanismos, no livro de M. Boden [2], criando o paralelo entre criatividade e criatividade computacional. A definição usada neste caso para criatividade computacional é "O estudo e suporte, através de meios e métodos computacionais, de comportamento exibido por sistemas naturais e artificiais, que seriam considerados criativos se exibidos por humanos.", sendo um sistema criativo o qual demonstre tal comportamento [3]. Tem vindo a ser discutido a questão da criatividade computacional, desde M. Boden [2] até G. Wiggins [3], [4] numa tentativa mais concreta do que se pode considerar um sistema criativo.

Devido à existência de vários métodos de optimização de soluções em conjunto com um problema complexo, nem sempre é fácil afirmar qual o melhor método para o problema em causa. Para este projecto foi escolhido o algoritmo genético, que baseia o seu processo

evolutivo na evolução natural. Já foi aplicado com sucesso inúmeras vezes e inspira-se num processo biológico em que os indivíduos (modelos Lego) mais fortes são maioritariamente os vencedores num ambiente competitivo [5], [6].

A motivação deste trabalho de mestrado é a implementação com sucesso de um sistema onde esteja presente a criatividade computacional, de modo a se retirar mais algumas conclusões quanto à criatividade sem interacção humana, através da produção de algo novo, surpreendente e valioso. O facto de ainda haver muito por descobrir sobre criatividade computacional, leva a que este trabalho seja importante para se poder entender melhor o conceito e aprofundar os conhecimentos. A sua ligação com os modelos LEGO e os algoritmos genéticos, leva-nos para uma área onde existem alguns projectos com uma ou outra componente semelhante, mas nenhum que conjugue todas as áreas abrangidas.

Tendo por base o trabalho desenvolvido por G. Wiggins [3], onde é descrita e analisada uma estrutura inicial para descrição, análise e comparação de sistemas criativos e por M.Boden [2], a finalidade desta dissertação passou pelo desenvolvimento de um sistema gerador de modelos de Legos de forma criativa. Tendo como ponto principal a criatividade, o projecto passa por outras áreas chave, tais como algoritmos genéticos e de procura aplicados à criatividade, estruturação de informação em formato digital e comparação de imagens.

Existem actualmente vários sistemas onde a criatividade apresenta um papel decisivo:

- JAPE - Um sistema que gera humor através de inteligência artificial. É um programa para produzir enigmas e trocadilhos, baseado em nove sentenças na forma geral do tipo *What kind of X can Y*?. Gerou por exemplo o enigma *What kind of murderer has fibre? A cereal killer* [7];
- Copycat – Um sistema que procura analogias entre listas de letras do alfabeto. A procura da analogia é orientada pela teoria que a descoberta de uma nova analogia é tal e qual como descobrir algo de uma nova maneira [8];
- EMI (Experiments in musical intelligence) – Um programa que compõe música com os estilos de Mozart, Stravinsky, Joplin, entre outros. Por vezes compõe uma frase musical quase idêntica a uma assinatura que nunca foi feita [9].
- EvoCAD – Um mini programa CAD para criação de estruturas Lego em 2D, que combina um algoritmo evolucionário (onde pertencem os algoritmos genéticos) simples com uma simulação parcial de estática de peças LEGO, evoluindo projectos que saem do computador prontos a construir, feitos num simulador de física simplificado [10].

Temos também alguns exemplos de sistemas que implementam criação de estruturas Lego:

- Representation and Evolution of Lego-based Assemblies – Evolução de grafos de gramática de montagem de modelos LEGO utilizando algoritmos genéticos. Aplicação das técnicas de optimização do algoritmo genético *Messy* na evolução de montagens compostas por estruturas LEGO [11];
- Solving LEGO brick layout problem using evolutionary algorithms – Uso do algoritmo genético para resolver problemas de optimização combinatória difícil em planos de construção LEGO [12];

- Using Sketches and Retrieval to Create LEGO Models – Sistema para criar modelos LEGO baseado em esboços [13].

Como anteriormente ilustrado, não existe nenhum trabalho que faça uso de todas as componentes aqui apresentadas, tornando assim este projecto de mestrado inovador.

1.2. Objectivos

O principal objectivo deste projecto foi a criação de um sistema que cria modelos LEGO de forma criativa, a partir de uma base de dados de peças LEGO, utilizando para validação imagens oriundas do Google Imagens. Durante a execução de todo o processo, o sistema evolui uma população de modelos Lego através do uso de um algoritmo genético.

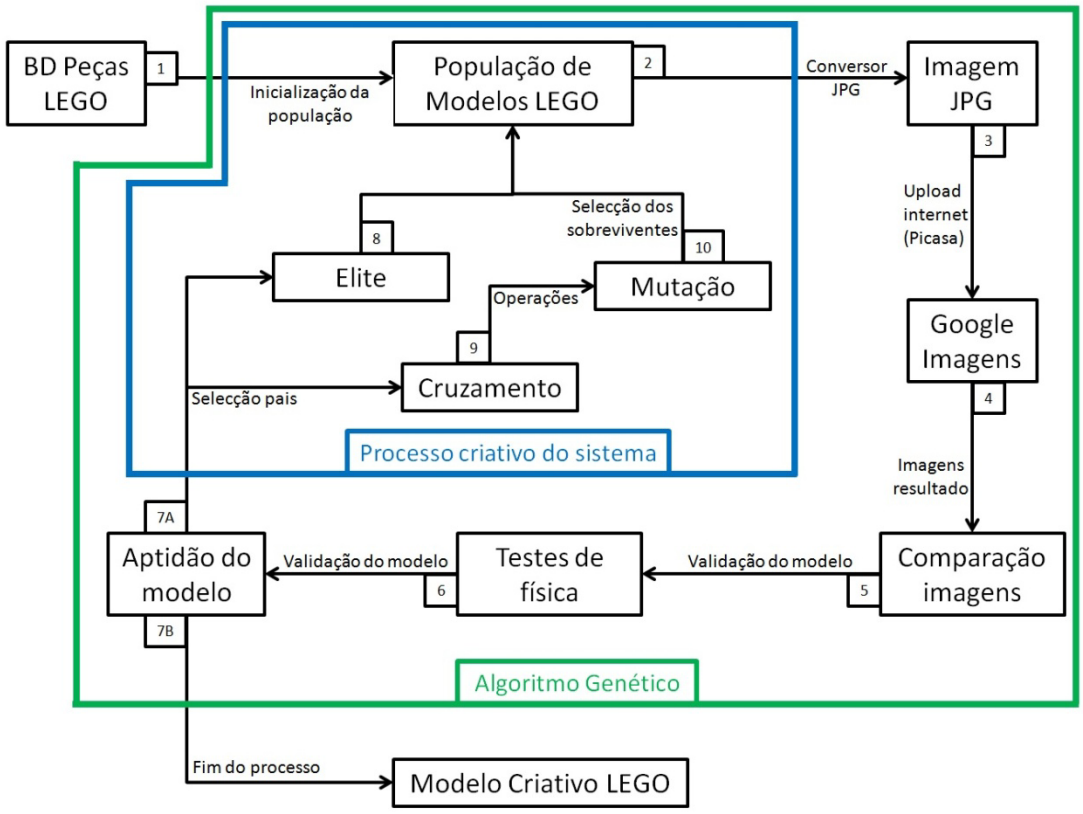

Figura 1 – Esquema de fluxo do sistema contendo todas as componentes, identificando a zona que se pretende criativa (a azul) e o algoritmo genético (a verde).

O sistema pode ser separado em várias componentes, como visível na Figura 1, tendo cada uma delas um ou vários objectivos que contribuem para a finalidade principal do sistema como um todo.

Criação de modelos LEGO de forma criativa (Objectivo principal – Obtido através da junção de todas as componentes do projecto).

Cada componente representada na Figura 1 tem o seguinte objectivo:

- **Ponto 1** – É a base de dados do sistema, onde é guardada toda a informação sobre as peças disponíveis no sistema;
- **Ponto 2** – Representa a população do sistema, um conjunto de modelos Lego criado inicialmente (população inicial) pelo algoritmo de procura alterado e que por sua vez é actualizada pela nova população sempre que passa uma geração do algoritmo genético. Aqui é utilizado o sistema de colisões de modo a validar a nova população do ponto de vista físico. Neste ponto é também guardada toda a população de modelos Lego em formato ficheiro ldr (explicado mais à frente), de modo a ser utilizado no ponto seguinte;
- **Ponto 3** – A imagem do modelo em formato jpg é obtida através da invocação de um executável, que transforma o ficheiro ldr do respectivo modelo Lego para imagem;
- **Ponto 4** – Neste ponto é onde se invoca o Google imagens em conjunto com a imagem do modelo (enviada para o Picasa) e se obtêm as imagens resultado para utilização na fase de comparação de imagens. A invocação utiliza um browser interno para processar o resultado e efectuar o download para o disco das respectivas imagens;
- **Ponto 5** – Esta componente é uma das duas partes de cálculo de fitness da população de modelos Lego, onde se efectuam as comparações pixel a pixel e de histogramas de cinzentos, com as imagens resultado obtidas anteriormente;
- **Ponto 6** – A segunda e última componente de cálculo de fitness passa pelos testes de física dos modelos;
- **Ponto 7** – Após os dois pontos anteriores temos todas as condições para calcular a aptidão (fitness) da população de modelos Lego. É também onde se decide se o sistema continua a executar (7A) ou termina (7B). Para o caso de já se terem atingido as condições de paragem do algoritmo genético (número de gerações) o sistema devolve o melhor modelo da população e termina. Caso não seja para terminar o sistema dá início à fase de geração de uma nova população;
- **Ponto 8** – A elite do algoritmo genético serve para reter os melhores indivíduos de uma geração para outra;
- **Ponto 9** – A fase de cruzamento da nova população selecionada através do método de selecção dos pais, serve para dar origem aos novos modelos através do cruzamento entre dois a dois modelos;
- **Ponto 10** – Na mutação, efectuada após o cruzamento, visa manter a diversidade entre a nova população gerada. Após esta fase, através da selecção dos sobreviventes em conjunto com a elite selecionada anteriormente, já temos a nova população da nova geração;
- **Zona azul** – É onde se encontra todo o processo do sistema que se pretende criativo, através da geração das várias populações de modelos Lego;
- **Zona verde** – Identifica todas as componentes do algoritmo genético, desde as usadas para cálculo de fitness da população (ponto 3, 4, 5, 6, 7) até às que compõem o ciclo.
- Agora inicia-se todo o processo novamente a partir do ponto 2.

A junção de todas as componentes, leva a um sistema criativo com verificação de quão criativo é o objecto criado. Através desta implementação, pretende-se avançar mais um pouco ao encontro da criação de um sistema totalmente criativo, sem interacção do utilizador. Também

se consolidam algumas das componentes de forma a ser possível a sua utilização em contextos mais relevantes. Destaca-se o uso inovador do Google Imagens, de forma a ser usado em aplicações futuras de procura de imagens. Temos também o uso do algoritmo genético, usado nas mais variadas situações, num contexto de modelação Lego com bons resultados. O uso de um sistema de física com sucesso, poderá levar a futuras implementações para validação do comportamento de modelos nas mais variadas situações. Por último, temos a metodologia usada para representação das peças Lego, que em conjunto com o LDraw (explicado mais à frente), poderá ser usada de forma a completar a mesma.

1.3. Planeamento

Para o desenvolvimento deste trabalho foram implementadas as seguintes etapas, visíveis no diagrama de Gantt (Figura 2). Algumas das etapas foram desenvolvidas em conjunto com outras, pois facilitaram o seu desenvolvimento e não existia qualquer impedimento para tal.

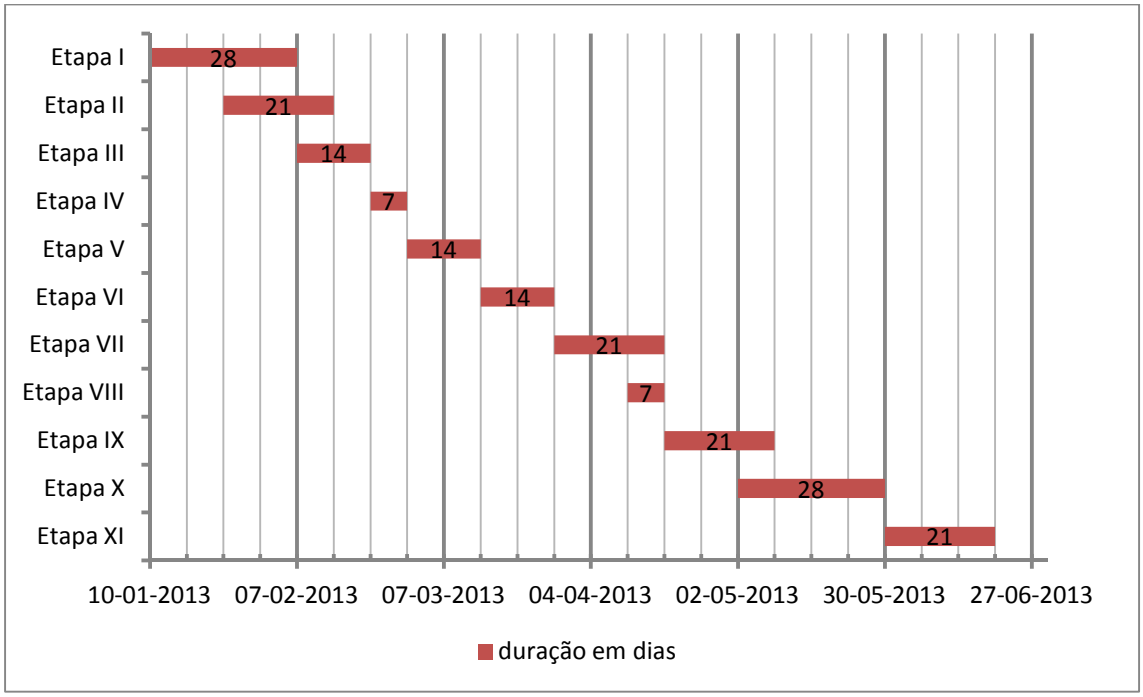

Figura 2 – Diagrama de Gantt da execução das etapas do projecto.

Em seguida identificam-se e descrevem-se as etapas em mais detalhe:

I. Criação de base de dados de peças LEGO. Construção de uma base de dados para guardar a informação de cada peça disponível no sistema para construção do modelo LEGO. A informação guardada passa por tamanho (eixo x, eixo y, eixo z), peso e encaixes;
II. Implementação de algoritmo de procura para montagem do modelo. Para implementação desta etapa, foi implementado um algoritmo simplificado de modo a que o sistema possa

gerar modelos de forma aleatória, escolhendo a peça a inserir e o seu local de destino no modelo de forma aleatória;

III. Implementação de um sistema de colisões para todo o modelo. De modo a que o modelo seja válido foi necessária a implementação de um sistema de colisões de forma a não haver sobreposição de peças. A implementação desta etapa passa por adicionar ao construtor de modelos uma validação de posição da peça;
IV. Conversão de modelos LEGO (formato "ldr") para imagem. Nesta etapa convertem-se os modelos gerados anteriormente (formato "ldr") em imagem (jpg) programaticamente através da execução de uma aplicação disponível na comunidade LDraw [14];
V. Envio de imagem para internet de modo a ficar disponível online para uso posterior. É necessária a disponibilização de uma imagem na internet de modo a poder ser usada posteriormente no Google imagens. Para esta etapa foi usado um site online de disponibilização de imagens do Google que apresenta uma API com as funções necessárias – Picasa [15];
VI. Obtenção de imagens semelhantes através do envio de imagem para o Google imagens. Esta etapa passa por enviar um endereço online de uma imagem para o Google imagens e obter as imagens resultado da primeira página, que correspondem às imagens mais relevantes/semelhantes para a que se enviou;
VII. Métodos de aptidão do modelo Lego. Esta etapa é composta pela comparação de uma imagem principal com as várias imagens obtidas anteriormente no Google Imagens, e pela validação da física do modelo num sistema de física. Este ponto permite identificar as semelhanças entre a imagem gerada do modelo Lego e as obtidas anteriormente, e efectuar testes físicos no modelo. No final obtém-se um valor para cada componente, de modo a ser usado para calculo da aptidão final do modelo;
VIII. Método que identifica a aptidão do modelo após toda a validação. Nesta etapa implementou-se um método que usa toda a informação obtida do modelo LEGO para gerar um valor de aptidão. Este valor é usado pelo algoritmo genético de forma a classificar cada modelo na população;
IX. Adaptação de todo o sistema ao uso do algoritmo genético, criando toda a estrutura necessária;
X. Definição e implementação dos métodos de cruzamento, mutação, selecção e elite do algoritmo genético;
XI. Execução do sistema como um todo e validação de resultados.

1.4. Estrutura do documento

Este documento está organizado da seguinte forma: no primeiro capítulo é introduzido ao leitor o contexto e motivação deste trabalho, a descrição detalhada dos objectivos atingidos e, por fim, o planeamento das etapas do projecto.

No capítulo seguinte, Estado da Arte, é realizada uma análise detalhada sobre a criatividade e a sua relação com os computadores, fazendo referência a alguns programas que usam criatividade. De seguida são introduzidos os sistemas LEGO, começando desde a sua criação até aos dias de hoje, mostrando também a relação do LEGO com o computador. Neste capítulo

são também identificados alguns programas na área do LEGO onde é feita uma análise mais detalhada de modo a se tentar perceber qual a melhor forma de implementação do projecto. Esta secção conta ainda com uma análise sobre os sistemas disponíveis para LEGO, incluindo a sua visualização. Na secção seguinte é introduzido e explicado o algoritmo genético, que dá a estrutura ao projecto e é usado para evolução da população de modelos. Em seguida, é referido o algoritmo de procura usado na construção dos modelos LEGO e o porquê da sua escolha. Nas duas últimas secções, são identificados alguns métodos de comparação de imagens e suas diferenças, terminando com alguns sistemas de visualização Lego.

No terceiro capítulo é descrita toda a arquitectura do sistema, onde se identificam todas as componentes implementadas e usadas pelo sistema. Começa com a descrição da base de dados implementada e dois exemplos de peças Lego guardadas. Em seguida explica-se o objecto principal do sistema, o modelo Lego. Na secção seguinte refere-se a componente de inicialização da população inicial do algoritmo genético. É também identificada a criação da imagem do modelo Lego, através do uso do respectivo ficheiro ldr. Nas duas secções seguintes, identifica-se o sistema Picasa, usado para disponibilizar as imagens dos modelos, e refere-se a obtenção das imagens resultado através do uso do Google Imagens. Nas próximas secções, descrevem-se os cálculos para a aptidão final do modelo Lego, passando pela comparação de imagens e simulação da física do modelo. De seguida, começa-se a descrever as várias restantes componentes pertencentes ao algoritmo genético, desde Elite e selecção dos pais, passando pelo cruzamento, mutação e sobreviventes, terminando com o fluxo principal do algoritmo. Na última secção deste capítulo, descreve-se a interface gráfica do sistema, explicando cada uma das suas componentes.

De seguida, no quarto capítulo são descritos os testes efectuados e analisados os resultados obtidos nos mesmos, começando por uma explicação geral das experiências efectuadas e respectivas configurações utilizadas. Na primeira secção, descreve-se em mais detalhe a fase um dos testes e quais as diferenças entre eles, analisam-se os resultados obtidos e efectua-se uma análise estatística dos mesmos. A fase um dos testes termina com uma breve conclusão sobre os mesmos, de forma a retirar algumas ilações e validar quais os configurações a usar na próxima fase. Na última e segunda fase dos testes, onde se testa o funcionamento completo do sistema, começa-se por dar uma breve explicação efectuando de seguida uma análise sobre os dados obtidos e termina com uma pequena conclusão sobre a segunda fase.

Em quinto lugar, temos as conclusões do projecto, onde são referidas as ilacções retiradas do projecto e dos respectivos testes efectuados.

No capítulo sexto, referem-se trabalhos e caminhos a seguir no futuro de forma a continuar a evolução deste projecto, podendo este evoluir em várias áreas bastante distintas.

No capítulo final temos a informação sobre toda a bibliografia usada, onde é possível aprofundar e validar os motivos que levaram ao desenvolvimento deste projecto.

2. Estado da Arte

Neste capítulo são clarificados os conceitos e tecnologias envolvidas no trabalho, de forma a contextualizar o leitor com a actualidade das várias componentes do projecto.

Inicialmente introduz-se o conceito de criatividade em sistemas de informação, onde são explicadas as três formas de criatividade existentes e são referidos alguns programas relacionados com criatividade. Seguidamente faz-se referência aos sistemas LEGO que existem actualmente e aos programas de visualização de peças actuais. Descrevem-se ainda os algoritmos de procura e genético para criação e evolução dos modelos criativos. Por fim, procede-se à análise de métodos de comparação de imagens.

2.1. Criatividade e os computadores

A criatividade é vista como uma característica da mente humana que não pode estar ligada aos computadores, sendo estes apenas máquinas que executam tarefas. No entanto, computadores e criatividade podem estar ligados, de modo a produzir o que se chama de "criatividade máquina". A criatividade é algo misterioso e difícil de explicar. Segundo o livro de M. Boden [2], a criatividade é "a habilidade de ter ideias ou artefactos que são *novos, surpreendentes e valiosos*". Podemos ter dois tipos de criatividade, a criatividade psicológica (criatividade-P) e a criatividade histórica (criatividade-H). Quando ocorre uma ideia e esta é nova apenas para o seu criador, mas já foi criada por outros, designa-se de criatividade-P. No caso da criatividade-H, um caso particular da criatividade-P, aplica-se quando a ideia que ocorre é inteiramente nova, nunca tendo sido criada por ninguém [2].

Podemos separar a criatividade de três formas diferentes, a primeira através da combinação de ideias familiares, tais como analogias ou combinação de imagens poéticas, que podem ser junções deliberadas ou, na maior parte das vezes, sem intenção. A segunda através da exploração, que consiste em explorar o espaço conceptual existente, ou seja, já existe mas ainda não foi explorado o suficiente para a encontrar. Temos como exemplo desta forma de criatividade a forma de escrever, teorias de química ou biologia. A terceira através da transformação, que consiste em transformar o espaço existente em algo novo e diferente. Temos como exemplos desta forma de criatividade a reorientação de estradas ou os artistas de rua [2].

Podemos afirmar que os computadores podem ser relevantes na área da criatividade, mesmo sendo difícil aceitar que uma máquina possa ser criativa. No entanto, é possível dizer-se que quem foi criativo foi o programador ou as pessoas responsáveis pela existência da máquina. Programando o computador para combinar, explorar ou transformar o espaço conceptual, consegue-se abranger todas as vertentes da criatividade abordadas.

Criatividade computacional é uma área da inteligência artificial que se foca em construir software que exibe comportamentos que se exibidos em humanos são considerados criativos ou efectuam algum aspecto de tarefa criativa [16]. Pode-se apresentar uma taxonomia, baseada na análise do estado da arte da área, de forma a separar o software de criatividade computacional em quatro classes [16]:

- Ferramentas de suporte à criatividade – Sistemas que assistem o utilizador na criação do produto, ajudando-o a concentrar nos aspectos criativos da tarefa. Exemplo: Autodesk Maya [17];
- Criatividade ajudada por computador – Sistemas que promovem a criatividade dos utilizadores ajudando-os na criação da ideia. Exemplo: NEvAr [18];
- Software de análise de tarefas criativas – Sistemas que efectuam algum tipo de análise a tarefas criativas ou produtos criativos. Exemplo: FreashJam [19];
- Artista artificial – Sistemas que criam conceitos e produtos novos e valiosos que exibem um grau de autonomia comparável aos humanos. Exemplo: Não existem sistemas nesta categoria, apenas algumas contribuições aproximadas.

Até à actualidade já foram desenvolvidos vários sistemas que produzem de forma criativa nos três tipos de criatividade, sendo os do tipo exploração os que obtiveram mais sucesso. Como exemplo temos os seguintes programas:

2.1.1. JAPE

Sistema que gera humor através de inteligência artificial. Originalmente criado por Kim Binsted em 1981, com supervisão de Graeme Ritchie e Helen Pain no Departamento de Inteligência Artificial da universidade de Edimburgo. Conta com várias versões desde a sua criação até aos dias de hoje. É um programa para produzir enigmas e trocadilhos, baseado em nove sentenças na forma geral do tipo *What kind of X can Y?* [7].

Contém quatro tipos de regras que efectuam tarefas diferentes em todo o modelo [7].

- *Schemata* – Configurações de léxico de enigmas subjacentes;
- *Sentence Forms* – padrões de texto fixo com espaço para outras cadeias de texto inserido;
- *Templates* – Definição de condições para que determinados itens sejam inseridos em formas de frase;
- *SAD (small adequate description) Generation Rules* – Cria estruturas linguísticas abstractas a partir de léxicos.

Existe um *lexicon, um conjunto de lexemes*, sendo que um *lexeme é* um grupo de informação linguística sobre um nome, verbo ou adjectivo. Um *shema* contém algumas pré-condições

lexicais, que descrevem em termos das suas propriedades e relações, a configuração de *lexemes* que irão formar a estrutura central de um enigma. O *SAD generator* actua na informação fornecida por um *schema* convertendo os *lexemes* ou sequências de *lexemes* em *SADs*, que são uma descrição mais detalhada da estrutura linguística. Um *template* contém algumas condições que descrevem a estrutura interna da *SAD relations* que recebe um método de extracção de informação. A informação extraída é posteriormente emparelhada com o nome de uma forma de sentença e passada ao *generator* que executa a última fase da criação de texto [7].

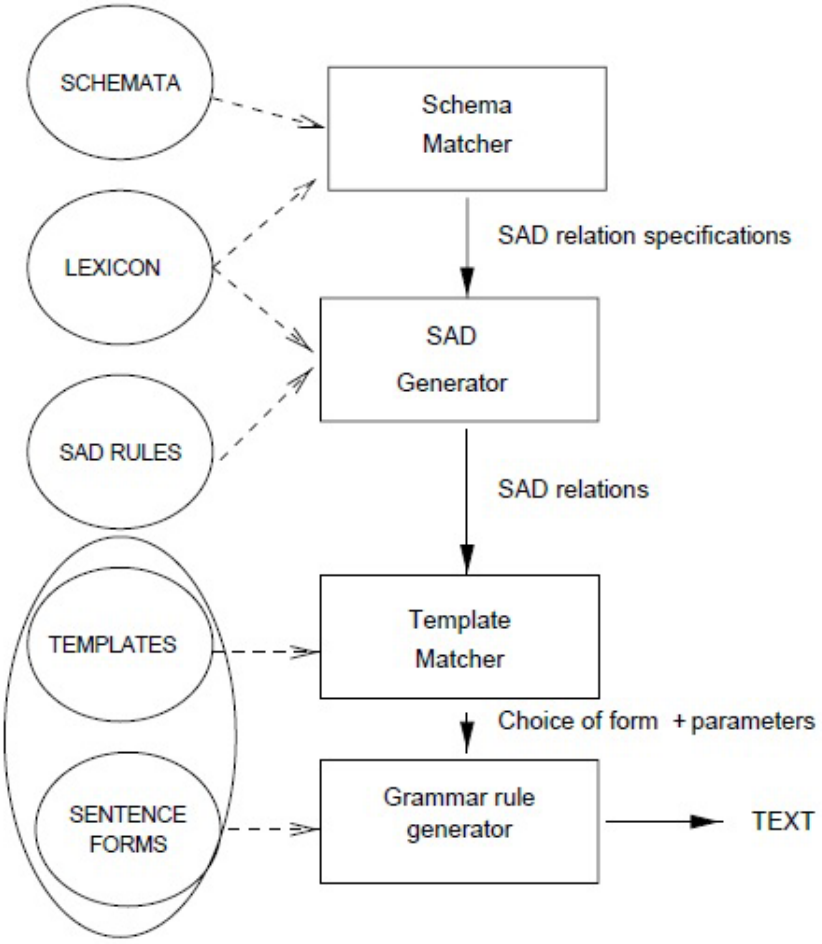

Figura 3 – Esquema geral do programa JAPE [7]**.**

Consiste num sêxtuplo de objectos, compostos por [7]:

- Um alfabeto A;
- Um *lexicon* LEX;
- Um conjunto SF de formas de sentenças sobre A;
- Um conjunto T de *Templates*;

- Um conjunto S de *schemata*, que contêm pré-condições definidas em termos de predicados definidos lexicalmente;
- Um *SAD generator* SG, com pré-condições de cada regra a ser definida em termos predicados definidos lexicalmente.

Alguns dos enigmas gerados pelo programa JAPE são referidos de seguida [7]:

- *What kind of murderer has fibre? A cereal killer* [4];
- *How is a nice girl like a sugary bird? Each is a sweet chick*;
- *What do you call a strange market? A bizarre bazaar.*

2.1.2. Copycat

Um sistema que procura analogias entre listas de letras do alfabeto, com base num experimento em não determinismo e analogias criativas. Foi criado por Douglas R. Hofstadter do Departamento de Ciência e Computação da universidade de Indiana no Laboratório de Inteligência Artificial no Instituto de Tecnologia de Massachusetts, na tentativa de se fazer um modelo fluido do processo de pensamento analógico. Foca-se em analogias num domínio idealizado, não escolhido arbitrariamente, mas emergido como produto de anos de tentativas na resolução da questão "O que faz uma boa analogia?". A procura da analogia é orientada pela teoria que a descoberta de uma nova analogia é tal e qual como descobrir algo de uma nova maneira [8].

O domínio concreto do sistema é o alfabeto, sendo cada entidade (letra) tratada como um objecto abstracto contendo uma ordem "platónica". Algumas definições de objectos são necessárias para se entender melhor o sistema [8]:

- *C-group (copy-group)* – Consiste num número não negativo de cópias de qualquer estrutura. Contém dois parâmetros, o número de cópias e a estrutura copiada;
- *S-group (sucessor group)* – Consiste num número não negativo de símbolos de letra na ordem da esquerda para a direita, cujos tipos ocorrem por essa ordem num alfabeto platónico;
- *P-group (predecessor group)* – é um grupo que contém o *S-group* de forma inversa.

Os problemas a ser resolvidos são todos do tipo: Se tivermos duas estruturas X1 (protótipo) e X2 (alvo), e modificarmos X1 de alguma forma, produzindo X1* (objectivo). Qual será "a mesma" modificação de X2? Isto é, qual será X2* (objectivo)? Chama-se *change* ou *visible action* à acção de remodelação a qual converte X1 em X1*, representada esquematicamente na Figura 4 [8].

```
visible action:     X1 (prototype)  ==>  X1* (result)
                    ------------------------------
                    X2  (target)    ==>  X2*  (goal)
```

Figura 4 – Esquema de acção de conversão do protótipo para objectivo no sistema copycat [8]**.**

A ideia de fazer "a mesma coisa" a um objecto completamente diferente, até mesmo com uma estrutura diferente é o que dá o nome ao projecto (Copycat) [8].

O sistema começa por uma fase de examinação da estrutura em causa, criando as ligações entre vizinhos (Figura 5), dando mais ênfase às ligações mais comuns e preenchendo os objectos definidos anteriormente [8].

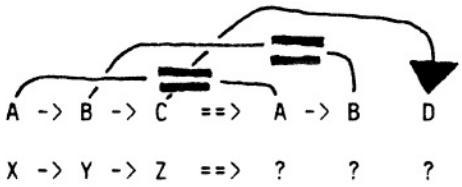

Figura 5 – Exemplo duma fase 1 do sistema Copycat [8]**.**

Após a primeira fase, chega-se à geração de regras, criadas através das ligações geradas na fase anterior. Algumas ligações e posições têm mais "peso" do que outras, começando o sistema a usar as de maior valor. Este "peso" é definido numa tabela de semântica. A fase seguinte é a fase crítica do sistema, onde se chega à comparação dos dois objectos (protótipo e alvo) de forma a validar se faz sentido aplicar a regra escolhida. Nesta última fase, a fase de execução o sistema conclui aplicando a nova regra e constrói a analogia final [8].

Temos um exemplo de uma interacção com o sistema para o alvo "PQRS" [8]:

ABC →ABD (*change* ou *visible action*)

PQRS → ???

Respostas possíveis:

- PQRT – Alteração da letra mais à direita pelo seu sucessor;
- PQRD – Alteração da letra mais à direita por "D";
- PQDS – Alteração da terceira letra por "D";
- PQSS – Alteração da terceira letra pelo seu sucessor;
- PQRS – Alteração de todos os "C"s por "D"s;
- PQRS – Alteração de todas as letras que ocorrem imediatamente à direita de "B" por "D";
- PQRS – Alteração de todas as letras que ocorrem imediatamente à direita de "B" pelo seu sucessor;

- QRST – Alteração de todas as letras depois de "B" num alfabeto platónico dos seus sucessores;
- PQD – Substituição de tudo após as primeiras duas letras por "D";
- PQBD – Alteração das duas letras mais à direita por "BD";
- ABD – Alteração da estrutura alvo por "ABD".

Cada uma das respostas anteriores vem acompanhada por um factor de pressão, que serve para decidir qual das respostas será usada [8].

2.1.3. EMI (Experiments in musical intelligence)

Programa que compõe música com os estilos de Mozart, Stravinsky, Joplin, entre outros, onde por vezes compõe uma frase musical quase idêntica a uma assinatura que nunca foi feita [4]. A ideia começou por criar um programa de computador que tivesse a percepção do seu estilo musical geral e que tivesse a habilidade de, em qualquer situação, permitir requisitar a próxima nota, próxima medida, próximas dez medidas e assim por diante. O programa desenvolvido por David Cope, professor emérito na Universidade de Santa Cruz, Califórnia, criava músicas de forma independente, conseguindo seguir as regras definidas para o estilo pretendido, mas as criações eram sem interesse e insatisfatórias [9].

A próxima alteração deu-se em usar músicas guardadas numa base de dados, visto todas as músicas terem um conjunto de instruções para criar músicas diferentes mas relacionadas com elas próprias. Estas instruções interpretadas de forma correcta podem levar a descobertas sobre a estrutura musical assim como novas instâncias da música estilisticamente fiel. A descoberta das referidas instruções foi baseada, em parte, no conceito de recombinação. Este pode ser definido como produtor de nova música pela recombinação de música existente numa nova sucessão lógica. Claro que combinação de partes de músicas necessita de uma extensa análise detalhada e uma recombinação cautelosa de forma a ser efectiva [9].

Pode-se separar o programa EMI em três princípios básicos [9]:

- Desconstrução – Analisar e separar em partes;
- Assinaturas – Manter o que significa estilo;
- Compatibilidade – Recombinar em novos trabalhos.

As novas descobertas efectuadas pelo programa levam às mais variadas reacções, desde encanto e provocação até irritação e terror nas pessoas que ouviram [9].

2.1.4. SWALE

Sistema que permite construir novas explicações para um dado conjunto de circunstâncias. A investigação no projecto SWALE foi conduzida por Alex Kass, David Leake e Chris Owens [20].

O projecto SWALE explora o raciocínio baseado em casos (CBR – Case-based reasoning) como base para criatividade. Neste caso, a criatividade vem da recuperação de conhecimento que

não é rotineiramente aplicado à situação e usa-o de uma nova maneira. Dependendo do processo de recuperação e adaptação usado, este pode obter soluções em qualquer sítio num espectro de criatividade, variando de reaplicações directas do conhecimento antigo até opiniões altamente inovadoras [1].

As questões principais são como retirar conhecimento apropriado para uso em novidade e como adaptá-lo de modo a atender novas circunstâncias. Dependendo do processo de recuperação e adaptação usado, o sistema pode sugerir soluções ao longo de um espectro de criatividade, desde reaplicações de conhecimento até visões completamente novas [20].

O uso de uma explicação que não se aplica ao caso concreto, dá-nos uma nova perspectiva da situação, levando a muitas questões tais como: Qual a explicação a devolver, que artimanha aplicar e durante quanto tempo aplicar a artimanha. São necessárias heurísticas, de forma a ter um mapeamento da explicação em causa, estratégias de artimanha que consigam fazer uma revisão significativa e heurísticas que permitam saber quando manter viva uma hipótese inútil no momento. Para o bom funcionamento do sistema é necessário também que este tenha uma memória rica em explicações (uma boa base de conhecimento) [20].

Como exemplo temos um cavalo de corridas que morreu aos 3 anos de idade quando estava no auge da sua carreira, de modo a tentar perceber o porquê desta morte o sistema SWALE indicou várias respostas possíveis com base em eventos passados. Pode ter tido um defeito cardíaco que causou um ataque cardíaco. Pode a pressão de ser uma estrela o ter levado ao consumo de drogas e por consequência a uma overdose. As mais variadas explicações foram obtidas através do sistema SWALE, sendo umas mais lógicas que outras [20].

2.1.5. Conclusão

Na sequência dos sistemas referidos anteriormente, através de uma análise cuidada, verifica-se que esta tese de mestrado apresenta várias diferenças a nível de tentativa de implementação de criatividade computacional. Neste caso, usada numa tentativa de criar algo com impacto visual para o utilizador, criando um modelo Lego apelativo à vista com semelhanças visuais de algo já existente.

2.2. LEGO®

A marca LEGO foi criada em 1932, pela família Kirk Kristiansen e pertence actualmente ao LEGO Group, com sede em Billund na Dinamarca e da qual a família Kirk Kristiansen é maioritária. O principal objectivo é o desenvolvimento da criatividade das crianças através de jogos e aprendizagem, tendo actualmente brinquedos, experiências e materiais de aprendizagem em mais de 130 países [21].

Actualmente existem milhares de peças LEGO reais diferentes, existindo quase infinitas possibilidades de combinação entre elas, sendo possível criar modelos de todos os tipos. Com o evoluir dos tempos e dos computadores começaram a ser desenvolvidas algumas migrações

para computador das peças e modelos LEGO. Com a existência de peças LEGO virtuais torna-se muito mais fácil, simples e barato construir modelos e criar combinações nunca antes efectuadas.

Alguns exemplos de programas com base no LEGO:

2.2.1. EvoCAD

Um mini programa CAD para criação de estruturas Lego em 2D. Combina um algoritmo evolucionário simples com uma simulação parcial de estática de peças LEGO, evoluindo projectos que saem do computador prontos a construir, feitos num simulador de física simplificado [10].

As estruturas LEGO usadas são uma representação genética em árvore através de funções lista em *Lisp*, que têm uma ou mais peças base com zero ou mais descendentes ligados em diversos pontos. O algoritmo genético aplica mutação que pode modificar o tamanho e posição da peça e recombinação que permite o intercâmbio de subpeças entre os dois pais. A representação genética na Figura 6 consiste numa peça base de 6 posições posicionada em (1,1), com dois descendentes, uma peça de duas posições localizada em (-1,1) e uma peça de quatro posições em (2,1). Todas as coordenadas das peças descendentes são em relação ao seu parente [10].

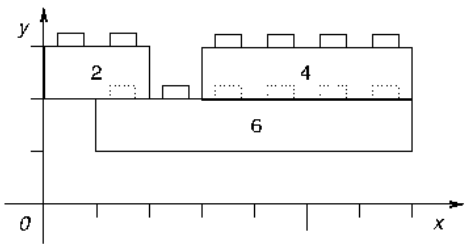

Figura 6 – Representação de uma estrutura LEGO no EvoCAD [10]**.**

Posteriormente as estruturas LEGO são simuladas num modelo simplificado de física onde as peças são tratadas como elementos rijos com as suas ligações a efectuarem um torque reactivo que compensa as forças externas até um certo limite. Após a avaliação de todo o sistema de forças consegue-se identificar se o sistema viola ou não alguma restrição [10].

Com a junção da representação genética e da simulação de física consegue-se evoluir as estruturas numa simulação. Adicionando uma função de fitness e aplicando um algoritmo genético em todo o sistema, basta esperar pelos resultados da evolução [10].

Foi também criado um mini sistema CAD (Figura 7) para desenho de estruturas 2D, de modo a permitir ao utilizador manipular as estruturas LEGO e testar a sua resistência gravitacional, usando um simulador estrutural simples. Este sistema usa todo o processo referido

anteriormente para evoluir soluções candidatas para o problema de desenho, podendo o utilizador definir novos objectivos ou alterar a solução [10].

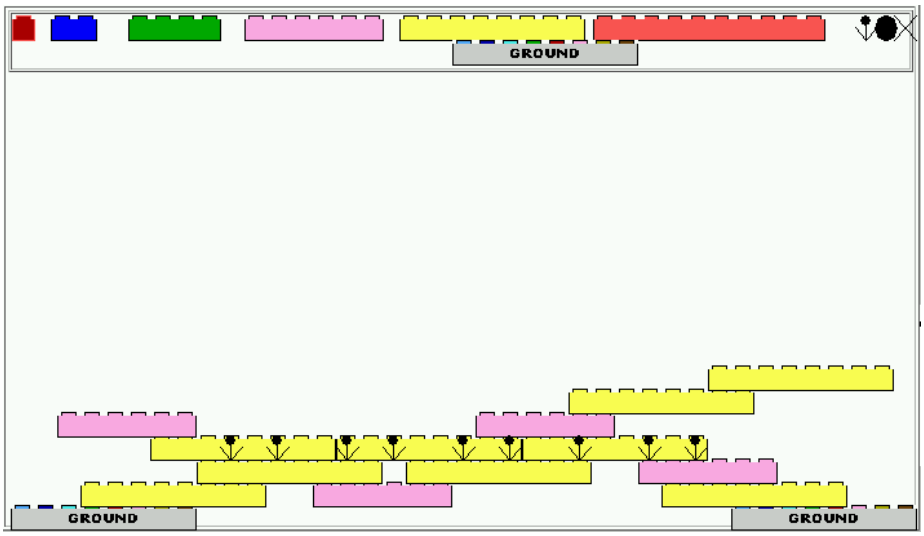

Figura 7 – Solução proposta pelo EvoCAD para as forças definidas inicialmente [10]**.**

O sistema global funciona da seguinte forma: é recebida uma estrutura que consiste numa ou mais peças, convertida posteriormente para uma representação genética que será usada como semente da população do algoritmo genético. A evolução tem início e os operadores de mutação e de recombinação são aplicados iterativamente de forma a fazer crescer e evoluir a população de estruturas. O simulador de física corre em todas as estruturas de modo a testar a estabilidade e resistência, de forma a poder calcular a aptidão da estrutura. A simulação pára quando todos os objectivos definidos forem satisfeitos ou o tempo definido acabar [10].

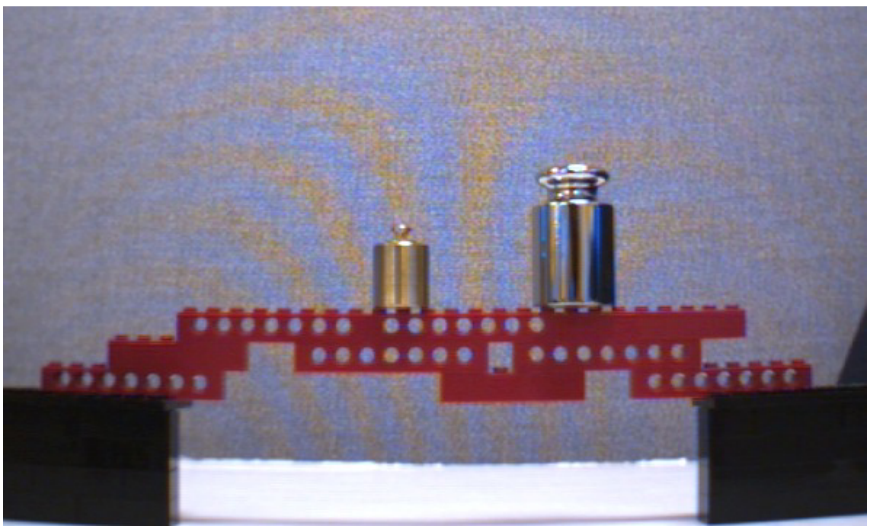

Figura 8 – Estrutura obtida através do EvoCAD construída com peças LEGO e testada com 250g de peso [10]**.**

Os resultados obtidos no sistema EvoCAD demonstram que este novo tipo de aplicação tem muito por onde evoluir, podendo o sistema ser não só uma ferramenta de simulação, mas também poder criar as suas próprias estruturas seguindo especificações do utilizador. Uma das maiores limitações do sistema é o uso de estruturas em árvore [10].

2.2.2. Sistema para representação e evolução de montagens baseadas em Lego

Pesquisa que deu origem a um sistema que evolui estruturas Lego num espaço de desenho não definido, usando algoritmos genéticos com uma representação baseada numa gramática de grafos. Criado por Maxim Peyakhov, Vlada Galinskaya e William C. Regli, no Centro de computadores Korma, na Universidade de Drexel [11].

Foi usada uma representação baseada em grafos de montagem, que apresenta mais benefícios quando comparada com as baseadas em árvore [22]. É uma estrutura mais expressiva e pode representar uma grande variedade de modelos LEGO incluindo mecanismos cinemáticos, bem como estruturas estáticas. Os nós do grafo representam diferentes elementos LEGO e as arestas as ligações entre elementos. Foi desenvolvida uma gramática de grafos de modo a completar a ausência de notação para descrever montagens LEGO válidas. Na Figura 9 pode-se observar um exemplo de uma estrutura LEGO e o respectivo grafo de montagem [11].

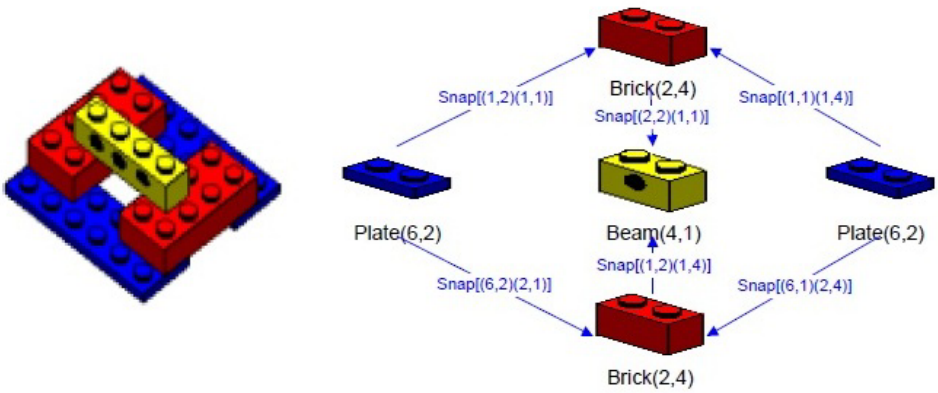

Figura 9 – Exemplo de estrutura LEGO com respectivo grafo de montagem [11]**.**

O funcionamento do sistema começa com a inicialização da população gerada de forma aleatória. São gerados dez nós de tipos aleatórios com dimensões aleatórias, em seguida são geradas de nove a treze arestas e posicionadas de forma aleatória tendo em conta que a estrutura tem de ser estável. Durante o processo de evolução é aplicado um operador de mutação que pode substituir elementos do tipo peça por outro do mesmo tipo de tamanho. O operador de recombinação é efectuado através do uso de dois operadores genéticos, corte e emenda, aplicado a dois cromossomas numa posição aleatória de cada. A função de avaliação usa os atributos da estrutura para poder chegar a um valor de aptidão final [11].

Um exemplo de evolução de uma estrutura estática após mil gerações, com valores predefinidos de mínimo de 10 unidades LEGO em cada um dos eixos (x,y,z), é mostrado na Figura 10 [11].

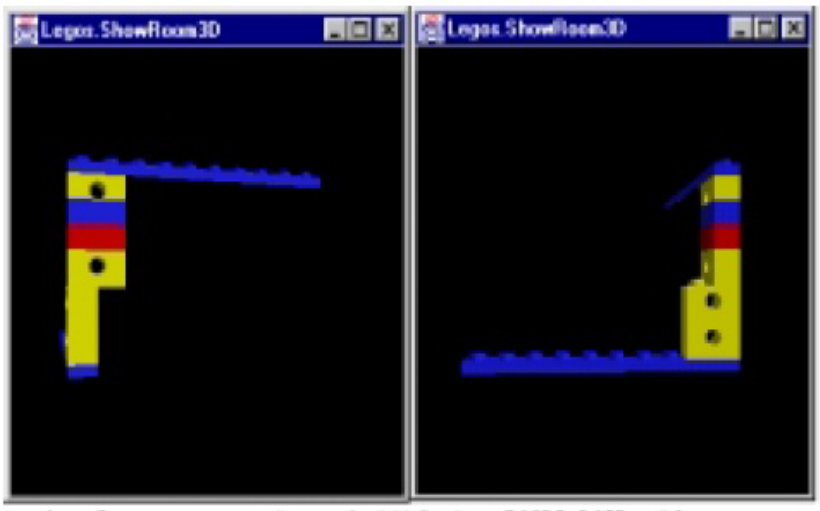

Figura 10 – Exemplo de estrutura após mil gerações com mínimos predefinidos de 10 unidades LEGO em cada um dos eixos [11]**.**

Uma das limitações é existir apenas três tipos de peças disponíveis no sistema, estando o sistema preparado para introduzir mais tipos. Outra das sugestões referida pelos autores para melhorar o processo de procura é existir uma inicialização guiada ou com semente [11].

2.2.3. LSketchIt – Using Sketches and Retrieval to Create LEGO Models

Sistema criado com o objectivo de implementar uma ferramenta simples e fácil de usar para criação de modelos LEGO, recorrendo ao uso de desenhos. Criado por Tiago Santos, Alfredo Ferreira, Filipe Dias e Manuel J. Fonseca, no departamento de ciência dos computadores e engenharia da Universidade Técnica de Lisboa [13].

O sistema combina interacção caligráfica, um solucionador de restrições e um mecanismo de reaquisição. Os desenhos são usados para especificar as dimensões da peça pretendida, para controlar a câmara e efectuar acções em elementos existentes. A componente de reaquisição usa as dimensões especificadas pelo utilizador no desenho, para efectuar uma busca na base de dados, mostrando como sugestão ao utilizador os que as satisfazem (visível na Figura 11). Após a escolha da peça pretendida, pode-se manipular a mesma no desenho devido ao sistema de restrições e ao alinhamento à grelha. O sistema detecta a existência de colisões e cria ligações entre peças ligadas podendo estas ser movidas como um todo. Apresenta um conceito de gravidade de modo a não permitir peças soltas no ar como na realidade [13].

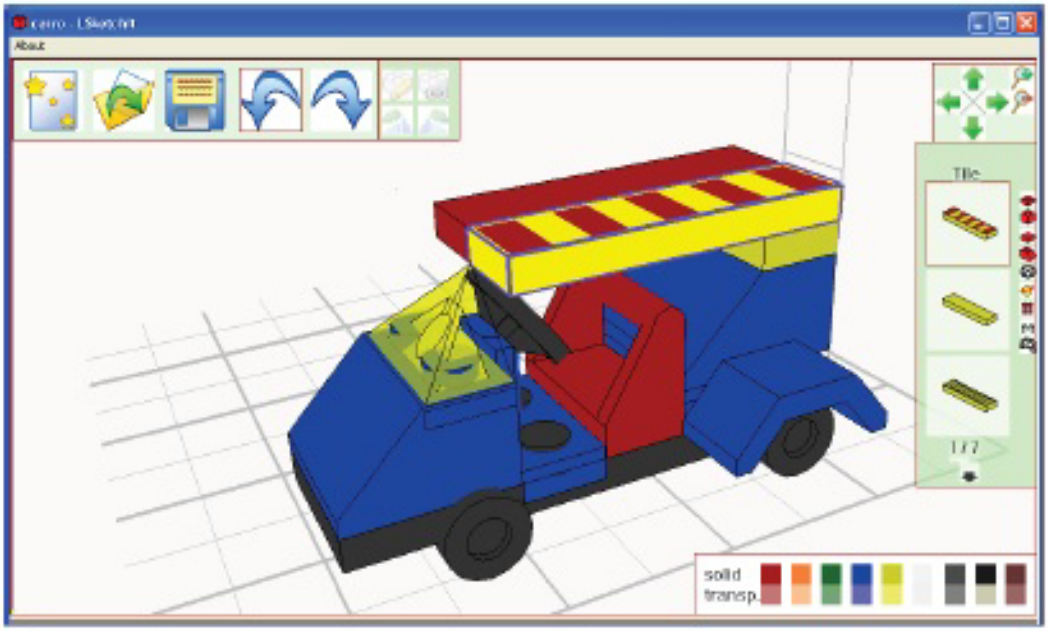

Figura 11 – Visão geral da aplicação LSketchIt, mostrando uma lista de sugestões à direita [13]**.**

Como base de peças foi usado o sistema LeoCAD, uma biblioteca de peças com código disponível que apresenta toda a informação da peça e permite a criação de modelos em 3D. O LeoCAD tal como o MLCAD são baseados na biblioteca LDraw, sendo a grande vantagem do LeoCAD a de já incluir uma biblioteca de peças já convertidas do formato LDraw, não tendo de se instalar a biblioteca LDraw [13].

Após uma comparação dos sistemas existentes na altura do início do projecto, foi detectado que os principais problemas eram a criação de modelos e pesquisa de peças. A implementação do projecto incidiu-se mais sobre os problemas referidos anteriormente (criação de modelos e pesquisa de peças). No final, após testes efectuados com utilizadores na aplicação desenvolvida e no LeoCAD, verificou-se que se conseguiu implementar de melhor forma as componentes referidas [13].

2.2.4. Conclusão

Após uma análise cuidada dos sistemas referidos anteriormente, verifica-se que este projecto de mestrado tem várias diferenças a nível do uso dos sistemas Lego. De entre as quais se destaca a estrutura desenvolvida para o modelo Lego, que apresenta ligações entre peças através dos seus encaixes, criando uma estrutura complexa de peças Lego. Esta estrutura foi considerada a mais adequada, pois está baseada numa estrutura em grafo e apresenta mais benefícios quando comparada com estruturas baseadas em árvore [22].

2.3. Algoritmo genético

É um método de procura baseado nos princípios da selecção natural e genética, que codifica as variáveis do problema de procura numa lista finita com uma certa cardinalidade, sendo essas listas referidas como cromossomas e os alfabetos como genes [23]. De modo a evoluir boas soluções e implementar selecção natural é necessário uma forma de distinguir uma boa solução de uma má solução, para tal existe uma função de fitness que determina a viabilidade da solução. Outro conceito importante nos algoritmos genéticos é a noção de população, um conjunto de soluções candidatas que tem um tamanho configurável. A descrição básica dos passos de um algoritmo genético passa pelo seguinte [23], visível na Figura 12:

1 – Inicialização da população inicial;
2 – Avaliação da população inicial;
3 – Selecção dos candidatos, normalmente com maior fitness, de modo a poderem ser evoluídos;
4 – Recombinação entre duas ou mais soluções dos candidatos seleccionados, de modo a evoluir as soluções (ponto opcional);
5 – Mutação dos candidatos seleccionados (ponto opcional);
6 – Substituição de indivíduos na população pelos evoluídos anteriormente;
7 – Repetição dos pontos dois a seis até ser atingido um critério de fim.

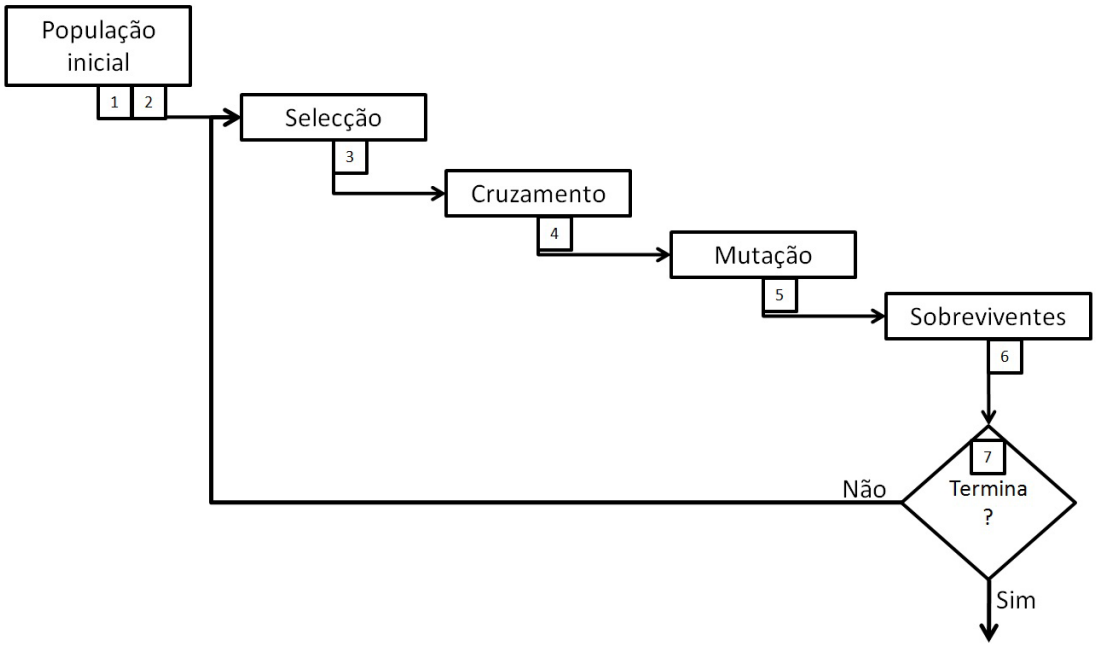

Figura 12 – Fluxo de um Algoritmo Genético.

O algoritmo genético que será usado foi anteriormente implementado nas mais variadas situações, sendo apenas necessário algumas adaptações ao problema (normal de acontecer em algoritmos genéticos) de modo a ser usado neste novo contexto. De seguida são apresentados alguns exemplos de situações em que foram usados:

- Roteamento de Veículos dinâmico usando algoritmos genéticos – Um problema que usa algoritmos genéticos, sendo neste caso usado para ajudar na decisão de rotas dinâmicas nos veículos em uso, de forma a reduzir os custos através dos modelos de distribuição e reduzir o tempo despendido [24];
- Problemas de planeamento florestal – Onde o algoritmo genético é usado neste caso com restrições de integridade para decisão. As variáveis de decisão representam alternativas de maneio (sequência de estados da floresta que ocorrem durante o horizonte de planeamento) e foram geradas variando-se a idade de corte e o regime de maneio, permitindo ciclos de um ou dois cortes [25];
- EvoCAD – Um mini programa CAD para criação de estruturas Lego 2D. A aplicação permite ao utilizador manipular estruturas Lego e testar a sua resistência gravitacional usando um sistema estrutural simplificado. Usa um algoritmo evolucionário que combina objectivos definidos pelo utilizador com simulação, para evoluir soluções candidatas para o problema de desenho. Os resultados de evolução são mostrados ao utilizador de modo a ser redesenhado até que uma solução satisfatória seja obtida [10];
- Representação e evolução de montagens baseadas em Lego – Sistema que evolui estruturas Lego com características pré-definidas. Para o seu desenvolvimento foi usada uma representação baseada em grafos e a sua codificação como um grafo de montagem com manipulação e evolução por algoritmos genéticos. Desenvolvimento de uma gramática de grafos para uso na representação de montagens Lego [11].

Depois de indicados alguns exemplos de uso do algoritmo, pode afirmar-se que este é usado nas mais variadas situações, não tendo sido encontrado nenhum exemplo do uso no contexto em que vai ser usado no sistema.

2.4. Algoritmo de procura

A palavra algoritmo tem uma longa existência, desconhece-se a sua invenção, mas tem como referência o século IX. Um algoritmo é um processo simples ou complexo composto por um conjunto de instruções, usado para efectuar certas tarefas, que podem ser ou não computacionais. Dois dos principais objectivos na criação de algoritmos são serem eficientes, de modo a não desperdiçarem recursos, e efectuarem com sucesso o propósito para que foram criados.

O algoritmo que será usado no programa para construção dos modelos é do tipo algoritmo de procura. Em termos gerais recebe um problema como entrada e retorna a solução para o problema na saída, procurando todas as suas soluções possíveis. Será alterado de forma a ser adaptativo na construção do modelo, fazendo ajustes às probabilidades de decisão na construção, conforme a qualidade obtida nos modelos anteriores. Foi escolhido este algoritmo por ser simples, relativamente fácil de implementar e de adaptar a componente dinâmica pretendida.

Alguns exemplos de uso de algoritmo de procura são referidos a seguir:

- Um Algoritmo de procura para planeamento de movimento com seis graus de liberdade – Utiliza o algoritmo de procura para ajudar a resolver o planeamento de instruções de movimento em robôs. É descrita uma implementação completa do algoritmo para resolução do problema, transformando o problema de planeamento de seis graus de liberdade num problema de navegação por ponto num espaço de configuração de seis dimensões [26];
- Um Algoritmo de procura simples e eficiente para estimativa do movimento por correspondência de bloco – Propõe-se um novo algoritmo de procura para redução de complexidade computacional para estimativa de movimento, de modo a substituir o algoritmo de procura em três passos (TSS) [27];
- Algoritmos de procura como Framework para optimização de combinação de medicamentos – Faz uso de um algoritmo de procura adaptado numa tentativa de inovação no uso do mesmo, de modo a optimizar combinações de intervenções terapêuticas. O algoritmo identificou correctamente combinações óptimas de quatro medicamentos usando apenas um terço dos testes efectuados em experiências biológicas de medição e restauro do declínio com a idade na função cardíaca e em capacidade de exercício em *Drosophila melanogaster*. Em experiências identificando combinações de três até seis doses, para morte selectiva de células de cancro humano, os algoritmos de procura adaptados resultaram num enriquecimento significativo de combinações altamente selectivas em comparação com pesquisas aleatórias [11].

Após a pesquisa de onde resultaram os exemplos anteriores do uso de algoritmos de procura, podemos afirmar que não foi encontrado nenhum algoritmo de procura utilizado no contexto onde este trabalho se insere. O que comprova a usa novidade no campo da criatividade.

2.5. Comparação de imagens

Na área de comparação de imagens existem várias abordagens distintas de modo a validar se são idênticas e quais as diferenças entre elas. Faz-se referência a algumas técnicas de comparação utilizadas em vários trabalhos de investigação:

- Transformada de Fourier (FFT) – Serve para classificar a frequência do conteúdo espacial da imagem, diferenciando por exemplo uma paisagem natural de um rosto ou cidade, tendo um FFT diferente em cada caso. Serve para classificar o tipo de imagem ou parte dela. É mais usado em pré-processamento de reconhecimento de imagem [28];
- Algoritmo SIFT (Scale-invariant feature transform) – Algoritmo que transforma uma imagem numa colecção de vectores de característica locais. Cada um desses vectores deverá ser distintivo e invariante em qualquer escala, rotação ou translação da imagem. Na implementação original, esses vectores podiam ser usados para encontrar objectos distintos em imagens diferentes [29];
- Histogramas de imagem – Registam a composição geral de cores numa imagem. São abundantemente usados na recuperação de conteúdos de imagens devido à sua eficiência e robustez. No entanto, apenas por si só, não chega para comparar imagens. Duas imagens

completamente diferentes podem ter histogramas iguais devido à sua composição geral de cores da imagem [30];

- Comparação pixel a pixel – É a comparação mais simples que existe, pois compara cada pixel de uma imagem com o da mesma posição numa outra imagem. Tem alguns problemas que podem ser facilmente ultrapassados com algumas adaptações, por exemplo, caso as imagens sejam de tamanhos diferentes ou tenham muitos pixéis diferentes, não é produzida muita informação sobre a diferença.

No sistema a implementar será usada uma junção de histogramas com comparação pixel a pixel adaptada, de forma a reduzir as desvantagens individuais de cada um dos métodos.

2.6. Sistemas e visualização

Neste capítulo explicam-se dois dos sistemas que existem actualmente para LEGO e as várias formas de visualização existentes para as peças/modelos criados virtualmente.

Podemos separar os sistemas LEGO em dois grandes tipos. Os que têm código fonte proprietário, ou seja, não têm disponível o código fonte/formato e são de direito exclusivo do produtor e os sistemas com código fonte aberto, que têm o seu código disponível para os utilizadores, permitindo o seu uso, cópia ou a sua alteração.

A empresa LEGO Group tem um sistema proprietário, o LDD – LEGO Digital Designer, que pode ser utilizado gratuitamente, dando a possibilidade de criar modelos LEGO com a maior parte das peças LEGO existentes na realidade. É um sistema de visualização com formato proprietário que permite criar as mais variadas combinações de peças e posteriormente partilhar o modelo criado na internet ou abrir modelos criados por outras pessoas. Permite também usar os modelos virtuais para construir o modelo na realidade, tendo a informação do número de peças necessárias de cada tipo [21]. O formato usado internamente não está disponível ao utilizador, não sendo possível o seu uso para este trabalho.

Para o tipo de código fonte aberto temos disponíveis várias opções, sendo o sistema mais usado o LDraw, que é um sistema composto por vários programas grátis de criação e modelação 3D para LEGO usando o formato LDRaw. É um formato aberto criado em conjunto com o programa original por James Jessiman, que morreu em 1997 ficando a manutenção e extensão assegurada pela comunidade LDraw, contando com muitas contribuições de utilizadores ao longo dos anos [14]. Permite a criação de modelos e cenários virtuais LEGO e pode ser usado para documentar modelos criados fisicamente, criar instruções de construção, renderização de imagens 3D e efectuar animações. Tem uma completa base de informação disponível sobre o formato e vários programas de apoio na construção de peças/modelos LEGO.

3. Arquitectura do sistema

Neste capítulo são descritas todas as componentes implementadas no sistema, onde se explica também o seu funcionamento.

3.1. Base de dados

A base de dados definida para o sistema, onde são guardados os detalhes das peças Lego disponíveis no sistema, tem a seguinte estrutura:

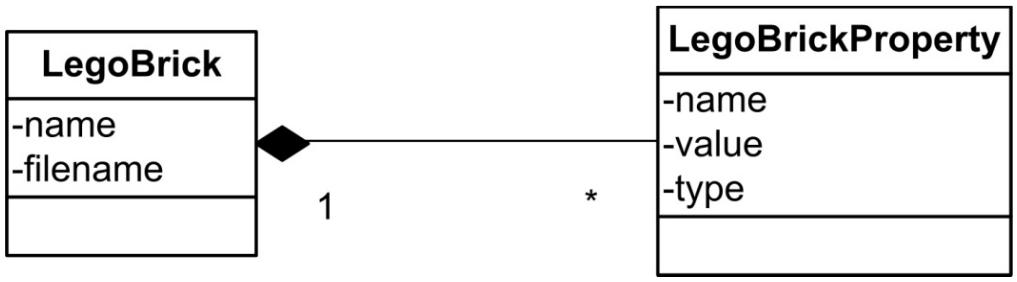

Figura 13 – Diagrama de classes da base de dados.

A Figura 13 representa o diagrama de classes da base de dados, composta por duas entidades que vão corresponder a três tabelas físicas. Na classe "LegoBrick" guarda-se a informação geral da peça (nome da peça e nome do ficheiro LDraw, já referido anteriormente). Na classe "LegoBrickProperty" são inseridos os detalhes da peça em questão, sendo o seu uso do tipo chave – valor. A seguir são mostrados dois exemplos de peças inseridas. Todas as medidas são em unidade do LDraw (LDU – LDraw Units), que equivale aproximadamente a 0,4 mm cada.

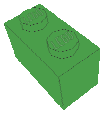

Figura 14 – Peça lego "Brick 1 x 2".

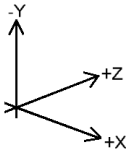

Figura 15 – Sistema de Eixos usado.

Figura 16 – Peça lego "Brick 1 x 6".

Tabela 1 – Registo na base de dados da peça lego "Brick 1 x 2".

Brick 1 x 2 – 3004.dat	
sizex	40
sizey	24
sizez	20
weight	2
stud1	-10;0;0
stud1	10;0;0
stud2	-10;24;0
stud2	10;24;0

Tabela 2 – Registo na base de dados da peça lego "Brick 1 x 6".

Brick 1 x 6 – 3009.dat	
sizex	120
sizey	24
sizez	20
weight	6
stud1	-50;0;0
stud1	-30;0;0
stud1	-10;0;0
stud1	10;0;0
stud1	30;0;0
stud1	50;0;0
stud2	-50;24;0
stud2	-30;24;0
stud2	-10;24;0
stud2	10;24;0
stud2	30;24;0
stud2	50;24;0

A peça "Brick 1 x 2 – 3004.dat", detalhada na Tabela 1, é composta pelas medidas 40 x 24 x 20 (X x Y x Z), tem peso igual a duas unidades, contém um conjunto de encaixes do tipo um (stud1 – visível na parte de cima da peça) e outro conjunto de encaixes do tipo dois (stud2 – encaixe na parte de baixo da peça). A peça "Brick 1 x 6 – 3009.dat", detalhada na Tabela 2, é composta pelas medidas 120 x 24 x 20 (X x Y x Z), tem peso igual a seis unidades, contém um conjunto de encaixes do tipo um (stud1 – visível na parte de cima da peça) e outro conjunto de encaixes do tipo dois (stud2 – encaixe na parte de baixo da peça). Ambas as peças têm como representação gráfica a Figura 14 e Figura 16 respectivamente, sendo a cor uma propriedade de visualização e não da estrutura. Cada encaixe tem posição relativa ao centro na imagem usada pelo LDraw. A Figura 15 representa o sistema de eixos usado pelo sistema LDraw.

3.2. Modelo Lego

O modelo Lego é a estrutura usada para representar um modelo Lego de forma virtual. É a principal estrutura no sistema e a que se optimiza através do uso do algoritmo genético, de forma a obter um modelo completo em todas as componentes presentes no projecto. É composta por uma lista de peças Lego, que por sua vez são compostas por uma lista de encaixes e várias propriedades que ajudam em todo o processo.

Os encaixes contêm informação sobre a sua posição em relação ao centro da peça, se estão livres ou ocupados e têm uma ligação à peça a que pertencem.

As peças são compostas pelo nome da peça no sistema LDraw, que serve para construir o ficheiro ldr, e uma lista de encaixes. Incluem também a posição da peça no modelo (coordenadas x;-y;z), a sua dimensão (x;y;z), peso e cor.

3.3. Algoritmos Genéticos – População inicial

A população inicial corresponde à parte de inicialização de um algoritmo genético. Neste caso é criada uma população de N (configurável) modelos Lego, de forma aleatória, usando o algoritmo de procura simples com alguns parâmetros e filtros configuráveis.

Parâmetros e filtros configuráveis:

- Número de peças no modelo;
- Ligação a peça já existente (probabilidade);
- Similaridade eixo X;
- Similaridade eixo Y;
- Similaridade eixo Z;
- Similaridade número de encaixes;
- Similaridade cor;
- Similaridade peso.

O algoritmo começa por escolher um número aleatório entre zero e um, de forma a escolher se insere uma peça noutra já existente (caso existam), ou se adiciona ao nível do solo. No caso de ser para introduzir uma peça ao nível do solo apenas vai ter em conta as peças disponíveis na base de dados, vai escolher uma de forma aleatória e tentar inseri-la num espaço aleatoriamente onde não colida com nenhuma peça já existente no modelo. Caso seja para introduzir numa peça já existente, vai aleatoriamente escolher uma peça do modelo com base nas probabilidades configuradas para os filtros. Nesta situação o algoritmo tem em conta que as posições a inserir têm de ser onde existam encaixes livres e não colidam com nenhuma peça já existente. Os encaixes são efectivados através da ligação entre encaixes, podendo percorrer todo o modelo através dos seus encaixes.

3.4. Ficheiros ldr e criação de imagens

Para a execução de todo o processo é necessária a criação da imagem do modelo de forma a ser usada mais à frente no processo. Começa-se por passar o modelo para ficheiro no formato ldr, escrevendo um ficheiro ldr, contendo as peças, as suas posições, cor e respectivo ficheiro LDRaw. Este ficheiro serve para visualização e conversão do modelo através do uso de programas adequados (LDView e MLCAD). Neste caso usa-se o LDView para conversão para imagem através da execução pela linha de comandos. Dois dos parâmetros dessa execução são o zoom e a perspectiva do modelo (ambas configuráveis), que vão alterar a imagem de saída.

Exemplo de uma execução do LDView para conversão para imagem:

- C:\Program Files (x86)\LDraw\LDView\LDView.exe\ ldr_file -SaveSnapshot=jpg_name -DefaultZoon=0.95 –DefaultLatLong=30,45

Conteúdo de um ficheiro ldr gerado pelo programa para um modelo de 10 peças:

- 1 4 290 -72 270 1 0 0 0 1 0 0 0 1 30144.dat
- 1 0 50 -144 310 1 0 0 0 1 0 0 0 1 30144.dat
- 1 1 330 -216 270 1 0 0 0 1 0 0 0 1 30144.dat
- 1 7 300 -96 260 1 0 0 0 1 0 0 0 1 3622.dat
- 1 1 30 -24 300 1 0 0 0 1 0 0 0 1 6111.dat
- 1 10 320 -120 260 1 0 0 0 1 0 0 0 1 3622.dat
- 1 7 250 -24 300 1 0 0 0 1 0 0 0 1 3008.dat
- 1 6 150 -48 300 1 0 0 0 1 0 0 0 1 6111.dat
- 1 8 10 -72 330 1 0 0 0 1 0 0 0 1 30144.dat
- 1 6 340 -144 270 1 0 0 0 1 0 0 0 1 3002.dat

Figura 17 – Conteúdo de um ficheiro ldr contendo dez peças Lego.

Cada linha no ficheiro ldr apresentado na Figura 17 corresponde a uma peça Lego no modelo e tem uma estrutura na forma “Tipo – Cor – X – Y – Z – Matriz3x3 – FicheiroLDraw”. O Tipo é a informação sobre o tipo de linha no ficheiro (existem seis tipos, mas o programa apenas usa o tipo um) e a Cor corresponde à cor da peça (número inteiro positivo). Em seguida vem a posição (X Y Z) da peça em relação ao ponto central do modelo e os valores da matriz 3x3 que representam a rotação e escala aplicada à peça. Por fim vem o nome do ficheiro LDraw correspondente à peça em causa.

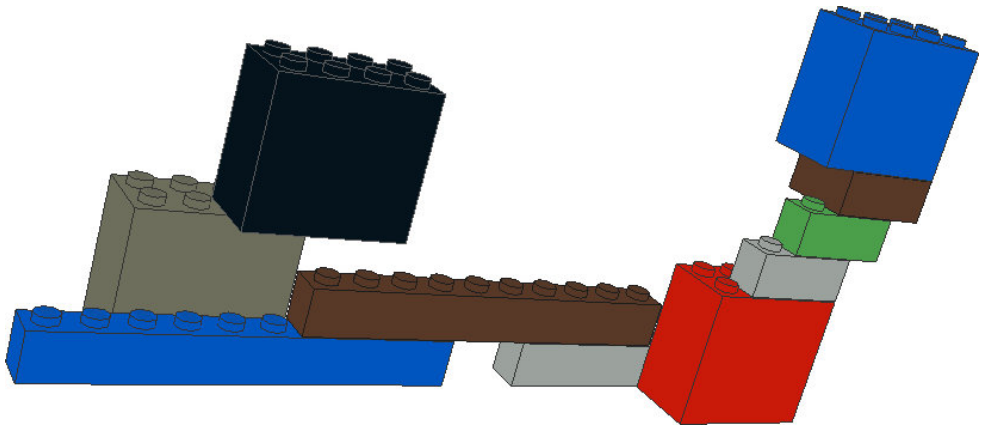

Figura 18 – Imagem do modelo Lego de exemplo.

Na Figura 18 é possível visualizar o modelo Lego de dez peças, correspondente ao ficheiro de exemplo apresentado anteriormente.

3.5. Disponibilização de imagem do modelo na Web – Picasa

Para a componente de comparação de imagens através do Google Imagens é necessário que a imagem a comparar esteja disponível na WEB. O Picasa é uma aplicação que faz parte da Google cuja função principal é organizar fotografias digitais e possui uma API de fácil acesso que permite o envio de imagens para álbuns online e posterior acesso através de um URL [15].

Para envio da imagem é usado o PicasaService (Google.GData.Photos.dll) juntamente com uma conta Google. É feita a autenticação com os dados de utilizador válidos e é enviada a imagem através do método "Insert", com a indicação do álbum onde inserir, os dados da imagem e o formato. Este método retorna uma entrada Picasa, que contém entre outros atributos o URL para a imagem disponível online que será usada noutra componente do projecto.

Este passo apenas é executado dependendo da configuração do sistema, pois pode-se configurar a comparação de imagens de três formas:

1. Inactiva – Não é executada a comparação de imagens e por isso não é necessário enviar a imagem para o Picasa.
2. Activa – É efectuada a comparação de imagens com os resultados do Google Imagens e por isso é necessário enviar a imagem para o Picasa.
3. Activa Pasta – Significa que é efectuada a comparação de imagens, mas com imagens guardadas previamente em disco e por isso não é necessário enviar a imagem para o Picasa.

3.6. Obtenção de imagens resultado do Google Imagens

Esta fase apenas é executada caso a comparação de imagens esteja configurada para comparação com imagens do Google Imagens. São obtidas as imagens resultado da primeira página (as mais relevantes) de resultados obtida através da invocação do Google Imagens com a imagem do modelo enviada anteriormente para o Picasa. No caso da comparação de imagem estar configurada para comparar imagens do disco, nesta fase é apenas adicionado ao modelo Lego o caminho do disco das imagens a comparar. Para a situação em que não existe comparação de imagens, esta fase não é executada. Na Figura 19 é possível visualizar o fluxo de todo o processo de obtenção de imagens resultado do Google Imagens.

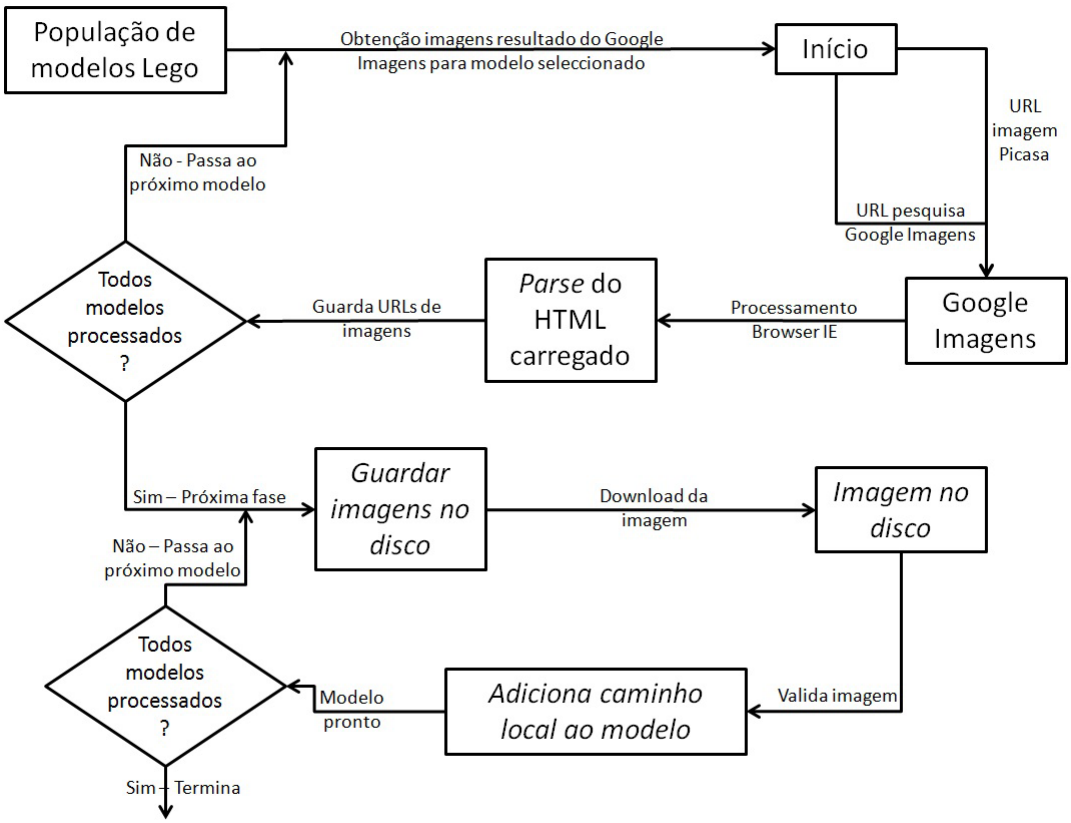

Figura 19 – Fluxo da componente de obtenção de imagens resultado do Google Imagens.

A obtenção das imagens resultado, é feita através da navegação num browser interno no sistema, contendo a URL de pesquisa do Google Imagens composta com a URL da imagem no Picasa. Após o carregamento do resultado da pesquisa são retirados todos os links para as imagens resultado que posteriormente são descarregadas para disco de forma a se poder efectuar a comparação num ponto mais à frente no sistema.

Exemplo de URL de invocação no Google Imagens:

- http://images.google.com/searchbyimage?image_url=URL_PICASA

3.7. Aptidão – Comparação de imagens

Esta componente diz respeito ao cálculo da aptidão duma parte do modelo, neste caso específico à comparação de imagens que é composta pela comparação pixel a pixel e comparação de histogramas de imagem. Esta componente apenas é executada caso a comparação de imagens esteja activa e é executada percorrendo todos os modelos e testando cada um deles de cada vez.

Na comparação pixel a pixel vai-se efectuar a comparação com as imagens resultado. Para isso a imagem resultado é convertida para o mesmo tamanho da imagem do modelo a comparar e

são percorridos todos os pixéis desde a posição [0;0] até à posição [largura; altura]. Quando os pixéis duma mesma posição são diferentes conta como uma desigualdade. No fim da comparação é retornado um valor entre zero e um, através da Equação 1, que representa a semelhança entre pixéis na mesma posição que foram encontrados. Caso seja retornado o valor um significa que a imagem é igual à comparada, se for devolvido zero representa uma completa desigualdade entre as imagens.

A comparação de histogramas é efectuada tendo em conta as imagens resultado, onde se comparam os histogramas de cada uma das imagens resultado com o histograma da imagem do modelo. Para este caso apenas são comparados os histogramas de cinzentos, em que a imagem foi convertida para cor cinzenta, onde se percorre cada posição do histograma do modelo e se compara com a respectiva posição do histograma da outra imagem. Em cada uma das comparações é calculada a diferença de valores a dividir pelo total de valores possíveis e guardado num vector de diferenças na posição X. No fim de se percorrerem todas as posições, soma-se o vector e divide-se pelo tamanho do mesmo, devolvendo um menos esse valor. Podemos confirmar na Equação 2 a explicação dada, que retorna assim um valor entre zero e um, sendo que um significa que são totalmente iguais e zero totalmente diferentes.

$$CI\ [0;1] = \frac{(nº\ total\ de\ pixéis - pixéis\ distintos)}{nº\ total\ de\ pixéis} \qquad \textbf{Equação 1}$$

$$CH\ [0;1]\ = 1 - \frac{\sum\left(\frac{abs(histograma\ imagem\ 1 - histograma\ imagem\ 2)}{total\ de\ posições\ no\ histograma}\right)}{nº\ posições\ do\ vector\ de\ diferenças} \qquad \textbf{Equação 2}$$

3.8. Aptidão – Física

Nesta componente é testada a restante parte da aptidão do modelo, através do uso de um sistema de física. É composto por dois testes de equilíbrio, com e sem aplicação de vento e chuva. Esta componente pode ser desactivada, não sendo testada a componente física do modelo, ou pode ser configurada para mostrar ao utilizador a simulação em tempo real. Esta última configuração apenas foi desenvolvida para ajudar no desenvolvimento do projecto e caso o modelo contenha mais de 50 peças não irá funcionar correctamente, devido a limitações da framework 3D usada.

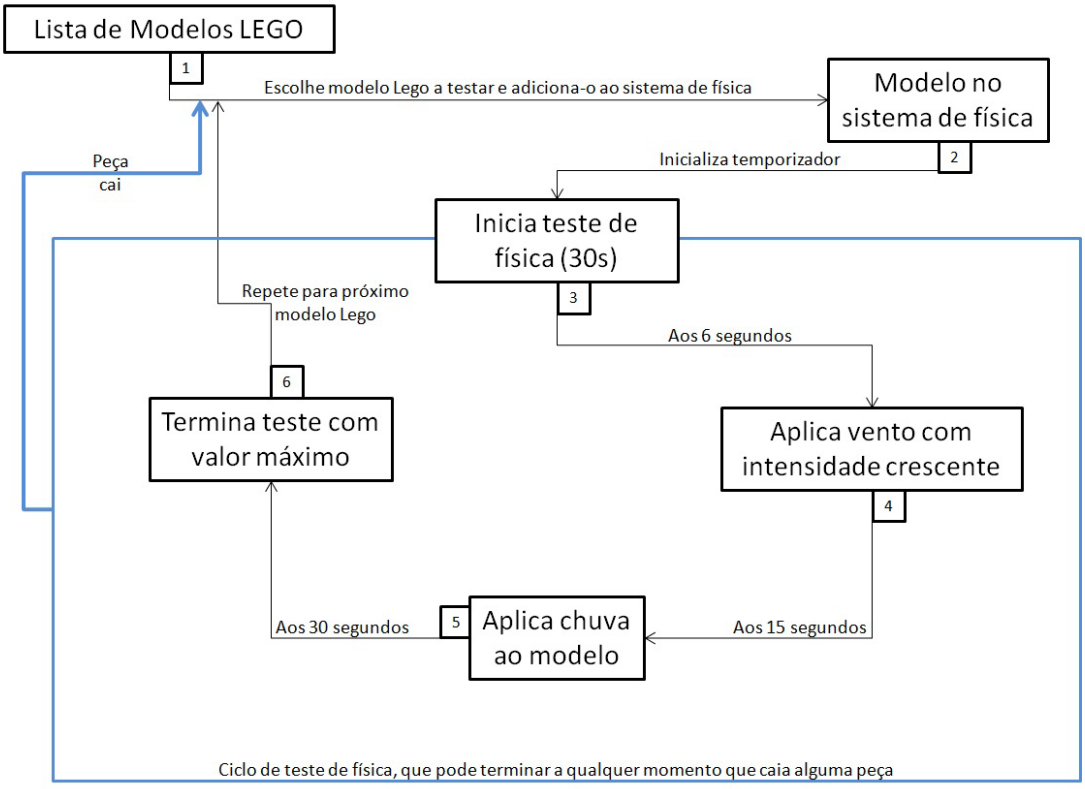

Figura 20 – Detalhe da fase de cálculo da aptidão da física do modelo Lego.

Esta fase começa por iterar sobre a lista de modelos a testar e vai efectuando os seguintes passos ilustrados na Figura 20:

1. Adiciona um chão (peça virtual) à listagem de peças do modelo a testar que servirá de base para o modelo no sistema de física;
   Criar o modelo – Passa por percorrer a listagem de peças Lego do modelo a testar e cria cada peça no sistema de física;
   Criar ligações entre peças – Este ponto permite que as peças no modelo estejam ligadas entre si como na vida real. São criadas as ligações entre peças, em cada um dos encaixes (studs);
2. Inicializa o temporizador de modo a testar cada modelo durante 30 segundos;
3. Começa por testar o equilíbrio do modelo durante os primeiros 6 segundos, caso este falhe, já não vai testar o restante, pois deixa de fazer sentido e volta ao início de forma a testar o modelo seguinte. Neste caso, todos os testes de física efectuados são considerados como falhados e têm valor zero para o cálculo final de aptidão;
4. Passados os primeiros 6 segundos, começa a aplicar vento ao modelo através da aplicação de uma força no eixo X em todos os centros de massa das peças do modelo, força esta que vai aumentando de 5 em 5 segundos até ao final do tempo;
5. Aos 15 segundos começa-se a introduzir uma simulação de chuva no modelo, de forma a testar o modelo quanto a condições atmosféricas reais (vento e chuva);

6. Caso o modelo chegue ao fim do tempo sem alguma das peças se mover ou cair, significa que este passou nos testes de simulação de física e vai contar com os seguintes valores para a aptidão final:
    - Estabilidade sem aplicação de vento ou chuva – 1;
    - Estabilidade com aplicação de vento e chuva – 1;
    - Máximo de força de vento aplicada – 2,5;
    - Máximo aplicado de chuva (apenas uma intensidade) – 1.

Os cálculos da aptidão final são explicados no capítulo seguinte.

3.9. Aptidão – Cálculo final

O cálculo final da aptidão (Equação 3) de cada modelo serve para conseguir comparar os modelos e poder aplicar o algoritmo genético. O valor final é a junção das aptidões apresentadas anteriormente, em conjunto com uma avaliação do número actual e inicial (NI) de peças no modelo, através da seguinte fórmula:

$$Aptidão\ final = CI \times 3 + CH \times 3 + E + EVC + V + C + N$$ **Equação 3**

- Comparação imagens (CI) → Melhor comparação pixel a pixel [0;1];
- Comparação histogramas (CH) → Melhor comparação histogramas [0;1];
- Equilíbrio sem vento ou chuva (E) → 0 (teve equilíbrio) ou 1 (não teve equilíbrio);
- Equilíbrio com vento e chuva (EVC) → 0 (teve equilíbrio) ou 1 (não teve equilíbrio);
- Máximo valor vento (V) → Máximo valor aplicado, neste caso vai de 0 a 2,5 (intensidade máxima de vento aplicada);
- Máximo valor de chuva (C) → Máximo valor aplicado, neste caso só existe 0 ou 1;
- Número de peças no modelo (N) → $\begin{cases} N \leq NI \rightarrow \frac{N}{NI} \\ N > NI \cap N - NI \leq NI \rightarrow \frac{NI-(N-NI)}{NI} \\ N < NI \times 0.1 \rightarrow -10 \\ N > NI \times 2 \rightarrow -10 \end{cases}$

A fórmula do N está feita de forma a promover no máximo o número de peças configurado, mas dando uma margem para cada um dos lados.

Os multiplicadores presentes no CI(3) e CH(3) foram escolhidos de forma a apresentarem um peso conjunto na mesma ordem da parte dos testes no sistema de física.

O valor mínimo que o valor de aptidão final pode ter é de -10 e o valor máximo possível é de 12,5.

3.10. Algoritmo Genético – Elite

Elitismo significa a retenção dos melhores indivíduos para a próxima geração sem modificações. Faz parte da fase de selecção do algoritmo genético. No projecto, o número de melhores indivíduos que passam para a próxima geração é configurável, tendo sido fixado o valor de um, ou seja, o melhor indivíduo passa sempre para a próxima geração. Garante-se assim que não se perde a melhor aptidão na população.

No início de cada geração, durante a selecção, os modelos são ordenados de forma descendente em relação à sua aptidão final, escolhendo-se sempre o primeiro modelo para passar à fase seguinte.

3.11. Algoritmo Genético – Selecção dos pais

Na fase de selecção dos pais, são seleccionados os modelos que passam à fase seguinte do algoritmo genético, de forma a serem aplicados os operadores de cruzamento e mutação. O método de selecção usado no projecto é a selecção por torneio, onde são escolhidos N modelos aleatoriamente, depois é escolhido o melhor modelo dos seleccionados para o torneio. O processo de torneio é executado várias vezes, até o número de pais seleccionados serem iguais ao tamanho da população, menos o número de elites seleccionados anteriormente. Para o projecto foi escolhido um torneio de três indivíduos, sendo este valor configurável nas definições do sistema.

3.12. Algoritmo Genético - Cruzamento

O cruzamento é um operador genético usado nos algoritmos genéticos que serve para variar a população através do cruzamento entre dois ou mais indivíduos. Para o projecto foi escolhido o cruzamento por um ponto variável, conhecido como cruzamento por “corte e emenda”, visível na Figura 21, em que se escolhe um ponto de corte variável e se cruza dois modelos através das suas peças, dando origem a dois filhos de tamanho variável.

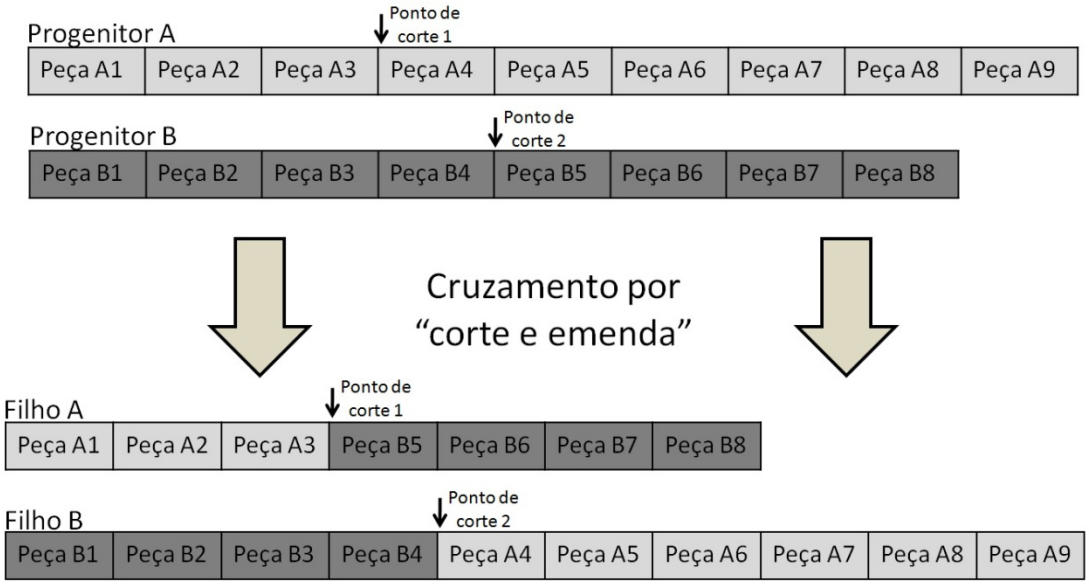

Figura 21 – Exemplo de cruzamento por um ponto variável (corte e emenda).

O método começa por receber a totalidade dos indivíduos a cruzar e uma probabilidade de cruzamento previamente configurada. Escolhe os dois primeiros indivíduos (I1 e I2) a cruzar e calcula um valor aleatório entre zero e um. Se o valor for superior à probabilidade configurada, os dois modelos são simplesmente copiados para a próxima geração. Caso seja inferior ou igual à probabilidade configurada, é iniciado o processo de cruzamento através do cálculo aleatório de um ponto de corte para cada (corte1 e corte2) modelo. Após a definição dos pontos de corte, começa por criar dois novos modelos (novo1 e novo2) e adicionar as peças aos mesmos da seguinte forma:

1. Adiciona as peças de I1 desde o início da lista de peças até ao ponto de corte1 ao modelo novo1;
2. Adiciona as peças de I2 desde o início da lista de peças até ao ponto de corte2 ao modelo novo2;
3. Adiciona as restantes peças de I2, desde o corte2 até ao fim da lista, ao modelo novo1;
4. Adiciona as restantes peças de I1, desde o corte1 até ao fim da lista, ao modelo novo2.

Nas fases 3 e 4, de cada vez que se adiciona uma peça, é testado se a peça a introduzir colide com alguma das já existentes e se tem encaixe na posição em que se encontra. Caso alguma das validações falhe (colida ou não encaixe na posição em que estava), a peça vai ser introduzida de forma aleatória no modelo através do algoritmo de procura inicial. Esta foi a forma encontrada para não gerar modelos inválidos através do cruzamento e não descartar simplesmente as peças que não encaixavam. Foi escolhido o cálculo de um ponto de corte para cada um, pois a lista de peças pode ter tamanhos diferentes.

Após ter sido percorrida toda a lista de modelos a cruzar o processo retorna com a nova lista de indivíduos.

3.13. Algoritmo Genético - Mutação

A mutação é um operador genético que visa manter a diversidade genética de geração em geração. Na Figura 22 é possível ver uma mutação a ocorrer num dos quatro objectos disponíveis na lista.

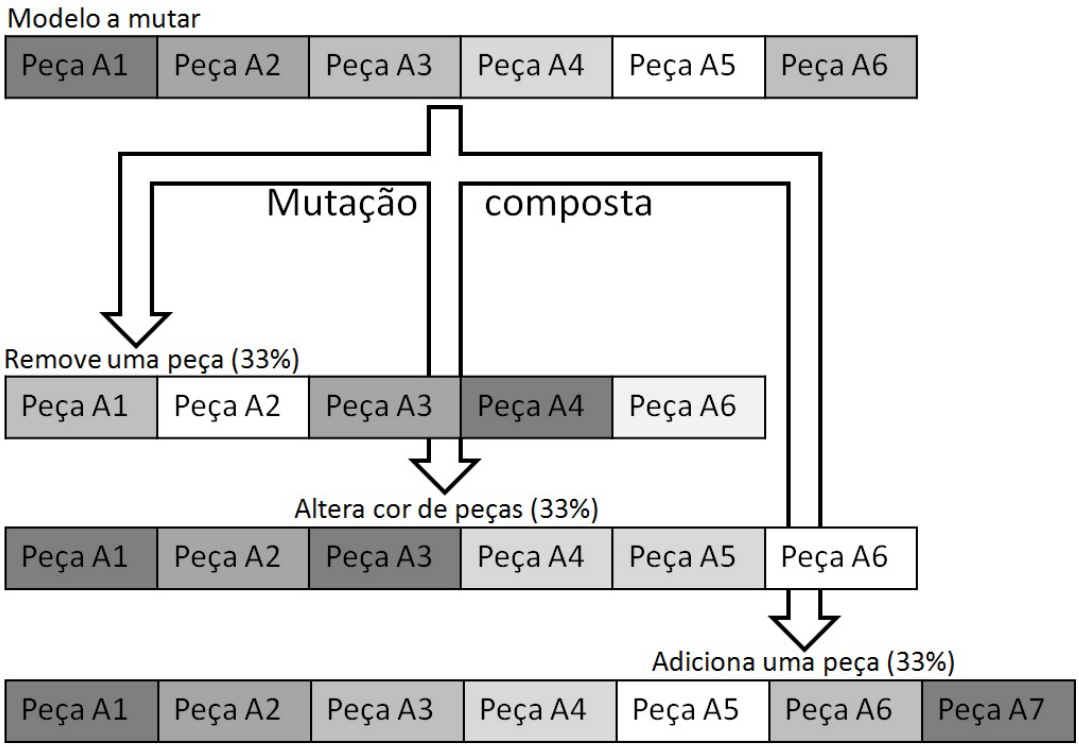

Figura 22 – Exemplo de mutação.

No projecto foi implementada uma mutação personalizada devido à complexidade dos objectos usados (modelos). O objectivo é conseguir chegar a todo o espaço de procura do sistema através de mutações sucessivas e passa pela junção dos seguintes três métodos:

1. Adição de peça ao modelo – Adiciona uma peça de forma aleatória ao modelo, tendo em conta os encaixes disponíveis e as colisões no sistema;
2. Remoção de peça do modelo – Remove uma peça aleatória ao modelo tal que ao se retirar esta peça não fique nenhuma outra inválida no modelo (suspensa no ar);
3. Alteração de cores – Percorre toda a listagem de peças e com uma certa probabilidade altera a cor da peça para outra aleatória.

O processo começa por receber um modelo e uma probabilidade P configurada previamente no sistema. Caso P seja maior ou igual a um número gerado aleatoriamente entre zero e um, vai ocorrer mutação, em caso contrário o modelo passa sem sofrer alterações. Para se decidir qual a mutação a aplicar de uma forma equitativa, cada uma delas tem um terço de hipóteses de ocorrer. Se o número gerado aleatoriamente for inferior ou igual a um terço, é executada a

adição de uma peça. No caso de ser maior do que um terço e inferior a dois terços é aplicada a remoção de uma peça e no restante é usada a alteração de cores do modelo.

Este processo é aplicado a todos os modelos previamente seleccionados como progenitores de modo a garantir diversidade da população.

3.14. Algoritmo Genético – Sobreviventes

Esta fase, referente ao algoritmo genético, é quase idêntica à fase de selecção dos pais, apenas difere na população a seleccionar. Para o projecto foi definido uma selecção de sobreviventes onde passam todos os filhos (escolhidos anteriormente), não havendo concorrência com os pais na selecção. A existência de elitismo garante a passagem do melhor indivíduo, os restantes advêm da selecção e aplicação dos operadores de cruzamento e mutação.

3.15. Algoritmo genético – Fluxo principal

Nas secções anteriores foram descritas todas as fases do algoritmo genético, que no projecto representa toda a parte de evolução dos modelos implementada. De seguida vai ser detalhado onde encaixa cada uma dessas componentes e todo o funcionamento global do algoritmo genético, que também pode ser visto na Figura 12.

A aplicação quando inicia a sua execução principal, começa por gerar uma população inicial de modelos Lego, com tamanho configurado previamente. Logo após essa fase, vai calcular a aptidão da população gerada, convertendo o modelo para ficheiro e posteriormente para imagem de modo a se poder comparar com as imagens alvo. Aplica ainda os métodos de fitness referentes à parte física do modelo e calcula a sua aptidão final. De seguida dá-se início ao ciclo do algoritmo genético, que vai ser executado em todas as gerações de forma a evoluir a população. Começa por escolher a elite para a próxima geração, efectuar o torneio para seleccionar os modelos a aplicar o cruzamento e a mutação e aplica-os. Após essa fase dá-se a escolha dos sobreviventes, de forma a escolher os modelos que passam para a próxima geração e, por fim, aplica-se o cálculo da aptidão dessa nova população. Caso ainda não se tenha atingido o limite de gerações configurado, é efectuado mais um ciclo do algoritmo genético.

3.16. Interface gráfica do sistema

A interface gráfica implementada é composta por um ecrã principal (Figura 23 e Figura 24), que contém uma zona que permite visualizar o modelo Lego, quando se está na fase de simulação de física. É composto por 3 botões, um para iniciar o processo, outro para aceder às configurações e o botão de sair do programa. Também tem disponível informação sobre em que estado se encontra todo o processo através da indicação da geração a executar, data de início e data da última mudança de fase. Tem uma zona de registo de actividade, onde ficam

registadas todas as fases por que vai passando, inclusive a aptidão final do melhor modelo em cada geração. Na parte de simulação de física pode ver-se o modelo em teste, o tempo de simulação, o vento e a chuva aplicados no momento.

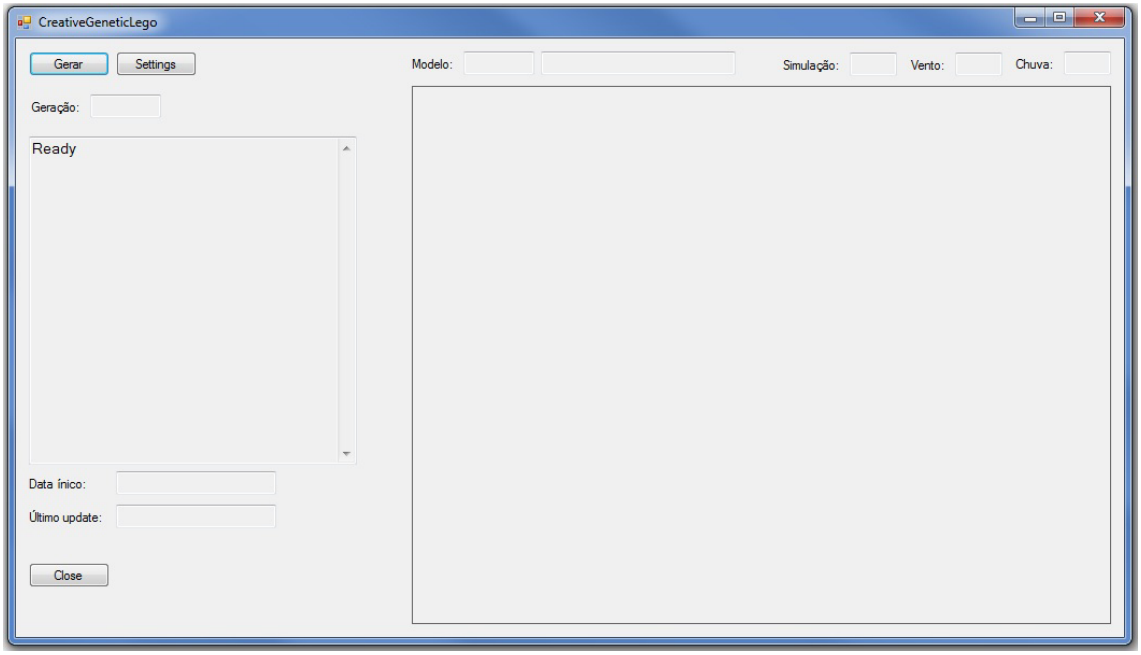

Figura 23 – Ecrã principal do programa.

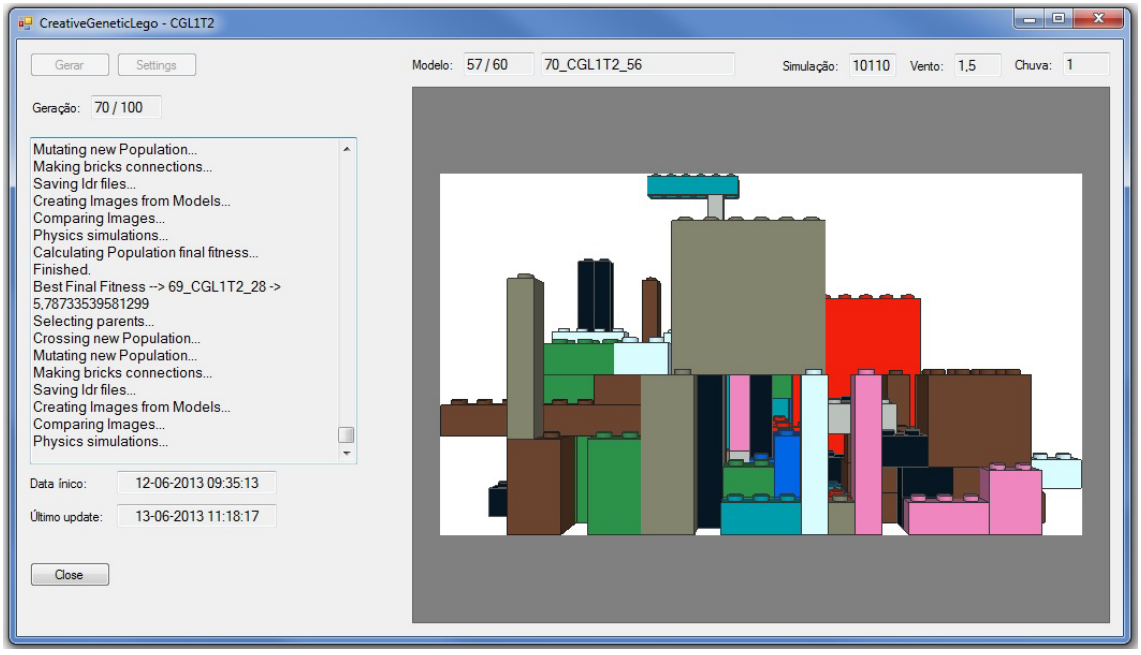

Figura 24 – Ecrã principal do programa em execução.

Existe ainda um ecrã de configurações, visível na Figura 25 e Figura 26, onde é possível definir os valores dos parâmetros alteráveis gerais, dos algoritmos e do sistema.

Configurações gerais do sistema (representado no rectângulo vermelho):

- Path – Local do disco onde o programa guarda os modelos e imagens criadas.
- Model name – Nome a aplicar aos modelos, que durante a execução é composto pelo prefixo do número da geração e sufixo de número de indivíduo dentro da população actual.
- Environment – Permite definir duas ligações à base de dados, neste caso uma "home" e outra "work", de forma a executar o programa em mais de um sistema sem ter de se alterar mais configurações nas conexões.

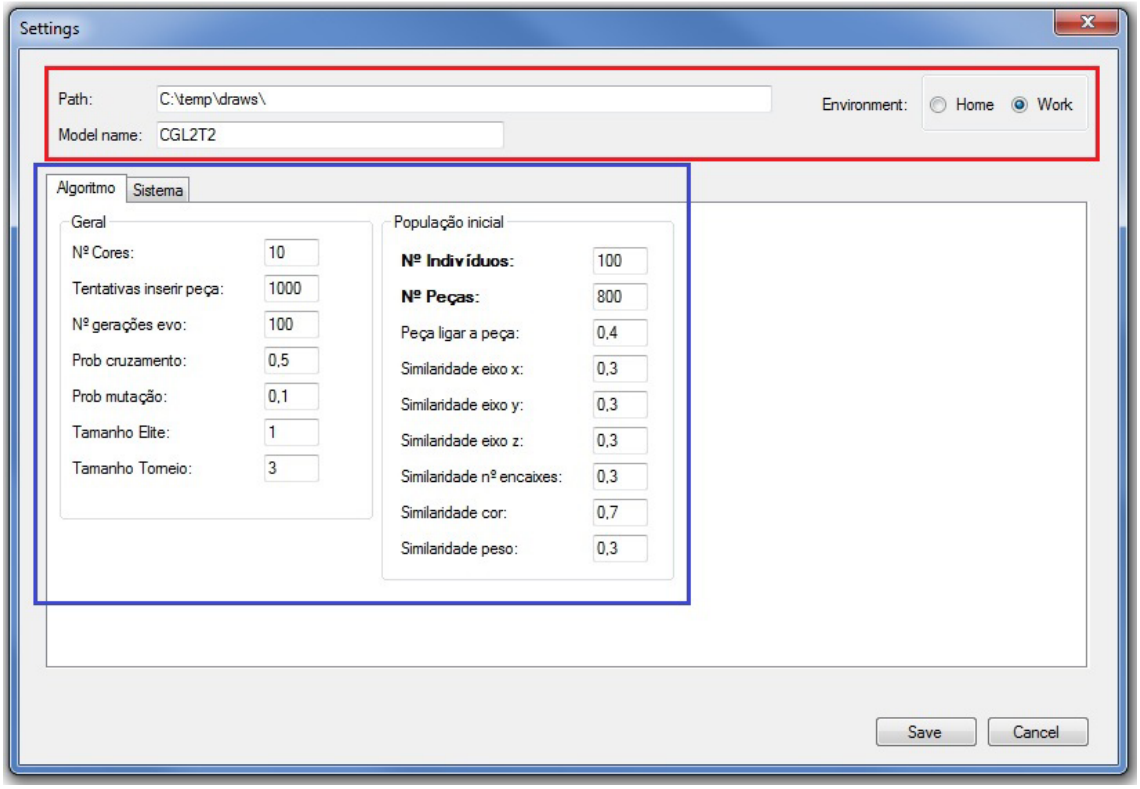

Figura 25 – Ecrã de configurações dos algoritmos. Configurações gerais a vermelho e configurações do algoritmo a azul.

Configurações dos algoritmos (Figura 25 – representadas no rectângulo azul):

- Nº Cores – Número máximo de cores do sistema LDraw a usar;
- Tentativas inserir peça – Número máximo de tentativas para inserir uma peça no modelo;
- Nº de gerações evo – Número de ciclos a executar pelo algoritmo genético;
- Prob cruzamento – Probabilidade de cruzamento;
- Prob mutação – Probabilidade de mutação;
- Tamanho Elite – Tamanho da elite;

- Tamanho Torneio – Tamanho do torneio usado para selecção;
- Nº Indivíduos – Tamanho da população;
- Nº Peças – Número de peças dos modelos a gerar inicialmente;
- Peça ligar a peça – Probabilidade da peça a inserir ser inserida numa já existente;
- Similaridade eixo x – Probabilidade da peça a inserir ser similar à de destino em relação ao eixo do X;
- Similaridade eixo y – Probabilidade da peça a inserir ser similar à de destino em relação ao eixo do Y;
- Similaridade eixo z – Probabilidade da peça a inserir ser similar à de destino em relação ao eixo do Z;
- Similaridade nº encaixes – Probabilidade da peça a inserir ser similar à de destino em relação ao nº de encaixes;
- Similaridade cor – Probabilidade da peça a inserir ser similar à de destino em relação à cor;
- Similaridade peso – Probabilidade da peça a inserir ser similar à de destino em relação ao peso.

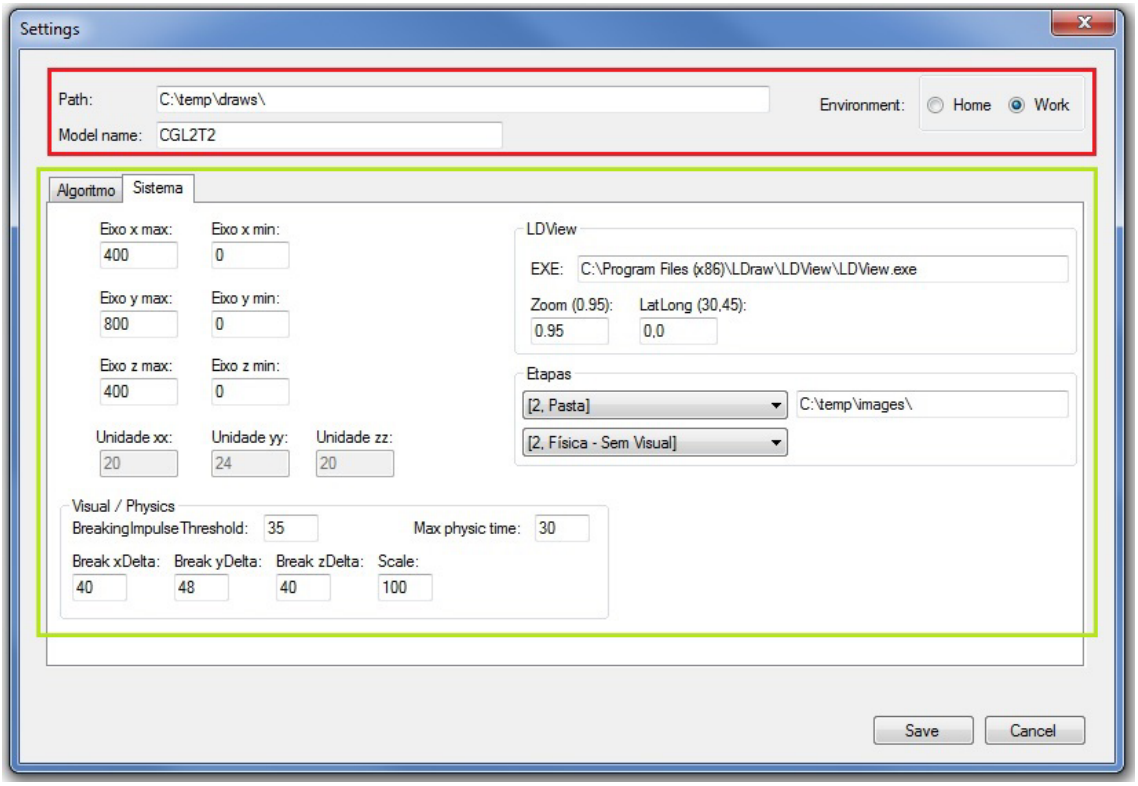

Figura 26 – Ecrã de configurações do sistema. Configurações gerais a vermelho e configurações do algoritmo a verde.

Configurações do sistema (Figura 26 – representadas no rectângulo verde):

- Eixo x max – Valor máximo possível ao inserir peças relativo ao eixo do X;
- Eixo x min – Valor mínimo possível ao inserir peças relativo ao eixo do X;

- Eixo y max – Valor máximo possível ao inserir peças relativo ao eixo do Y;
- Eixo y min – Valor mínimo possível ao inserir peças relativo ao eixo do Y;
- Eixo z max – Valor máximo possível ao inserir peças relativo ao eixo do Z;
- Eixo z min – Valor mínimo possível ao inserir peças relativo ao eixo do Z;
- Unidade xx – Valor da unidade em relação ao eixo do X;
- Unidade yy – Valor da unidade em relação ao eixo do Y;
- Unidade zz – Valor da unidade em relação ao eixo do Z;
- Breaking Impulse Threshold – Valor de quebra das ligações entre peças;
- Max physic time – tempo máximo para teste da física do sistema;
- Break xDelta – Valor máximo no eixo do X usado para detectar quebras no modelo;
- Break yDelta – Valor máximo no eixo do Y usado para detectar quebras no modelo;
- Break zDelta – Valor máximo no eixo do Z usado para detectar quebras no modelo;
- Scale – Escala usada para criar o modelo no sistema de física;
- EXE – Caminho no disco para o executável do programa "LDView" que efectua a criação da imagem do modelo;
- Zoom (0,95) – Zoom a aplicar na criação da imagem do modelo (valor por defeito);
- LatLong (30,45) – Valores de latitude e longitude para a perspectiva na criação da imagem (valor por defeito);
- Etapas – Configuração das duas etapas principais do sistema.
    - Comparação de imagens – 0: inactiva; 1:comparação com imagens do Google Imagens; 2:Comparação com imagens do disco (caminho para pasta local);
    - Testes de física – 0: inactivos; 1:activo com visualização em tempo real (apenas implementado para testes e uso com menos de 40 peças por modelo); 2:activo sem visualização em tempo real (apenas mostra a imagem do modelo).

4. Resultados

Neste capítulo são resumidas as experiências realizadas, fase um e fase dois, contendo uma breve discussão sobre as mesmas. A primeira fase serviu para se ajustar da melhor forma alguns dos parâmetros disponíveis e validar todo o processo, e a segunda fase para se testar o funcionamento completo do sistema.

Na primeira fase de testes foram experimentados vários valores de probabilidade de cruzamento e de mutação, tendo a comparação de imagens configurada para usar imagens em disco e não os resultados do Google Imagens. Foram utilizadas quatro imagens em disco e validação de física dos modelos Lego do sistema activa. Efectuaram-se doze execuções de cada uma das configurações, de forma a ter uma base sustentável para escolha dos melhores parâmetros para a fase seguinte.

A segunda fase de testes, onde já se encontra activa a comparação de imagens com imagens obtidas do Google Imagens, foi efectuada com as melhores configurações obtidas nos testes anteriores. Ficou definido uma probabilidade de 0,5 e 0,1 para o cruzamento e mutação respectivamente para a segunda fase de testes. O número de execuções para esta fase foi a mesma da fase anterior (doze execuções).

Ambas as fases foram executadas recorrendo ao uso de dois computadores distintos, onde correram várias instâncias do sistema, com as devidas configurações.

Em seguida referem-se as configurações usadas durante ambas as fases de teste, estando a negrito as que podem ser alteradas de teste para teste.

As configurações usadas para os testes foram as seguintes:

- Nº cores disponíveis – 10;
- Tentativa de inserir peça – 1000;
- Gerações – 100;
- **Probabilidade de cruzamento** – {0,1;0,5};
- **Probabilidade de mutação** – {0,1;0,7};
- Elite – 1;
- Tamanho do torneio – 3;
- Nº indivíduos na população – 60;
- Nº peças para o modelo – 800;

- Probabilidade de peça a inserir ligar a outra existente – 0,4;
- Probabilidade de peça similar eixo X – 0,3;
- Probabilidade de peça similar eixo Y – 0,3;
- Probabilidade de peça similar eixo Z – 0,3;
- Probabilidade de peça similar nº de encaixes – 0,3;
- Probabilidade de peça similar cor – 0,3;
- Probabilidade de peça similar no peso – 0,3;
- Valor máximo eixo X – 400;
- Valor mínimo eixo X – 0;
- Valor máximo eixo Y – 800;
- Valor mínimo eixo Y – 0;
- Valor máximo eixo Z – 400;
- Valor mínimo eixo Z – 0;
- Impulso para quebrar ligações entre peças – 35;
- Tempo máximo para teste de física – 30;
- Diferença entre posição original e actual de peça para quebra do sistema (eixo X) – 40;
- Diferença entre posição original e actual de peça para quebra do sistema (eixo Y) – 48;
- Diferença entre posição original e actual de peça para quebra do sistema (eixo Z) – 40;
- Escala para construção no modelo de física – 100;
- Caminho do executável para conversão do modelo para imagem – LDView.exe;
- Zoom na geração da imagem do modelo – 0,95;
- Latitude e longitude para geração da imagem do modelo – 0,0;
- **Etapa de comparação de imagem** – {0:inactivo; 1:comparação imagens do Google Imagens; 2:comparação imagens disco};
- **Etapa de validação da física do sistema** – {0:inactivo; 2:activo sem visual em tempo real};

4.1. Fase 1

A primeira fase de testes foi efectuada de forma a validar qual a melhor configuração dos parâmetros de probabilidade de cruzamento e de mutação testados. Nesta fase todos os restantes parâmetros foram constantes, sendo que a comparação de imagens foi efectuada com quatro imagens guardadas em disco, não usando o Google Imagens. Os dois valores de cruzamento foram escolhidos de forma a não prejudicar em demasia a população e não desactivar por completo a mesma. No caso da mutação escolheu-se o valor mais baixo para simular alguma diversidade e o mais alto de forma a testar este caso específico de Lego que poderia dar bons resultados [31].

Foram feitos os seguintes conjuntos de valores a testar:

- Teste 1 → Cruzamento = 0,5 e Mutação = 0,7;
- Teste 2 → Cruzamento = 0,5 e Mutação = 0,1;
- Teste 3 → Cruzamento = 0,1 e Mutação = 0,7;
- Teste 4 → Cruzamento = 0,1 e Mutação = 0,1;

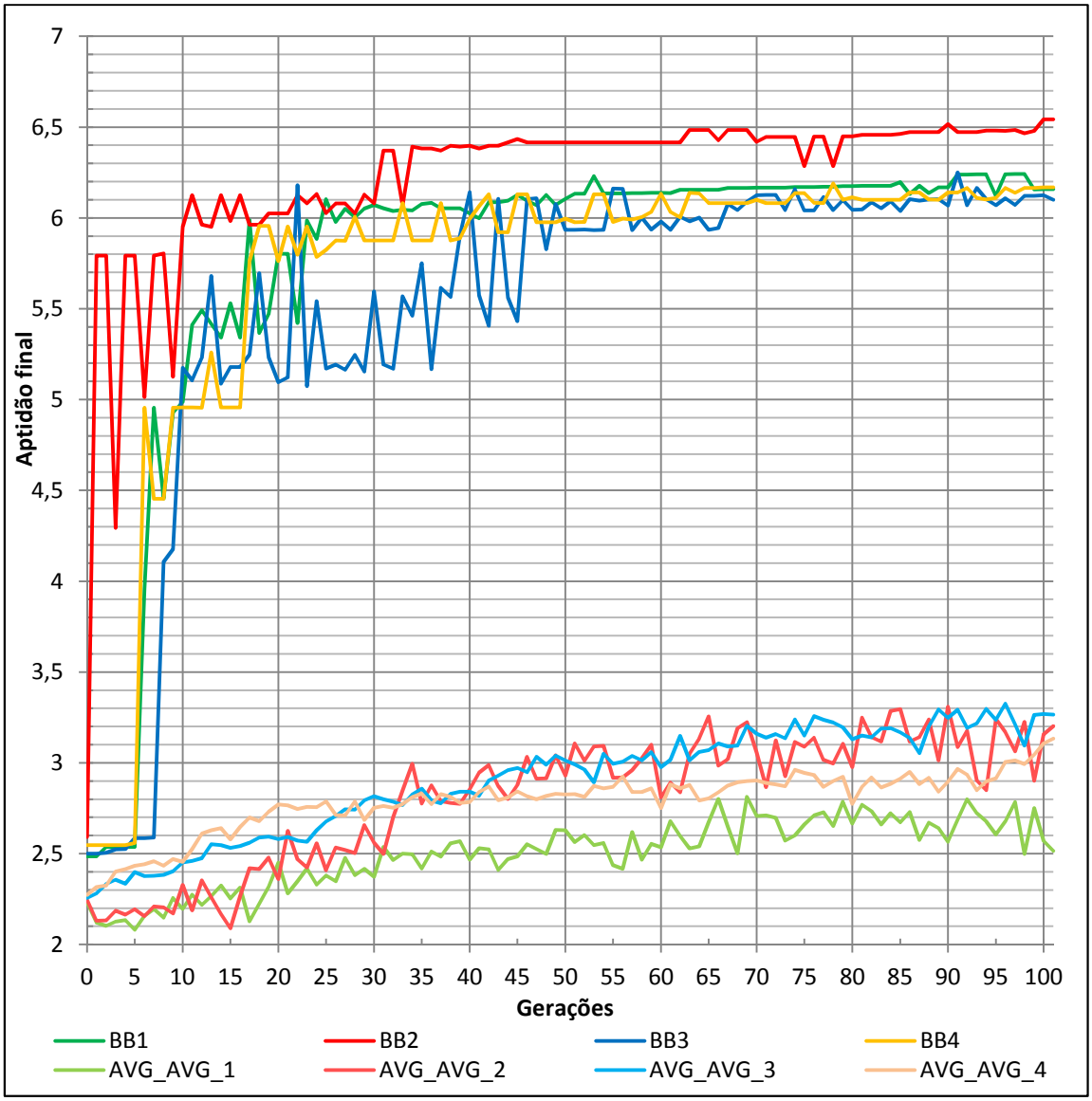

Figura 27 – Gráfico da primeira fase de testes, contendo informação do melhor dos melhores e média das médias das populações de cada teste.

Na Figura 27 pode-se observar a evolução dos quatro testes, obtida através do decorrer do algoritmo genético. A legenda "BBN" significa o melhor dos melhores em relação ao teste N, no caso da legenda "AVG_AVG_N" observa-se a média das médias da população do teste N. Nas médias não se consegue distinguir visualmente de forma objectiva qual o melhor conjunto, mas se observarmos no melhor dos melhores, verifica-se que o teste dois se destaca ligeiramente pela positiva.

Para melhor analisar os resultados, foram efectuados os testes estatísticos referidos de seguida.

4.1.1. Dados Paramétricos

O conjunto de dados obtido ser paramétrico é importante para a escolha do teste estatístico de grupo a aplicar para análise do conjunto de dados. Para os dados se considerarem paramétricos têm que cumprir os quatro seguintes requisitos: seguir uma distribuição normal, ser homogéneos, estar contidos num intervalo e ser independentes.

De seguida analisa-se o conjunto de dados quanto à sua distribuição normal, que serve para validar se estes possuem uma série de propriedades. Em caso afirmativo, estas propriedades, simplificam os cálculos no uso de alguns métodos estatísticos, fazendo com que alguns dos resultados sejam mais fiáveis e completos.

Para testar a normalidade dos dados usaram-se os testes de Kolmogorov-Smirnov e Shapiro-Wilk [32]

Tabela 3 – Testes de Normalidade Kolmogorov-Smirnov e Shapiro-Wilk para a primeira fase de testes.

Teste de Normalidade							
Valor analisado	Teste	Kolmogorov-Smirnov			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Melhor aptidão	1	,293	12	,005	,726	12	,002
	2	,328	12	,001	,564	12	,000
	3	,324	12	,001	,758	12	,003
	4	,402	12	,000	,673	12	,000

Dos dados apresentados na Tabela 3, concluiu-se que os dados não seguem uma distribuição normal, pois todos apresentam uma significância inferior ou igual a 0,05 (intervalo de confiança de 95%).

Do teste anterior, verifica-se que o conjunto de dados não pode ser considerado paramétrico, pois uma das condições essenciais falha (o conjunto de dados seguir uma distribuição normal). Com o resultado do teste anterior, já não é necessário efectuar mais testes de verificação dos requisitos paramétricos, pois uma das condições necessárias já falhou.

4.1.2. Teste estatístico de grupo

Devido aos dados não serem paramétricos, não é possível usar testes estatísticos de grupo para dados paramétricos. O teste não é tão fiável quanto o respectivo para dados paramétricos, mas é fiável o suficiente para retirar algumas conclusões sobre os testes.

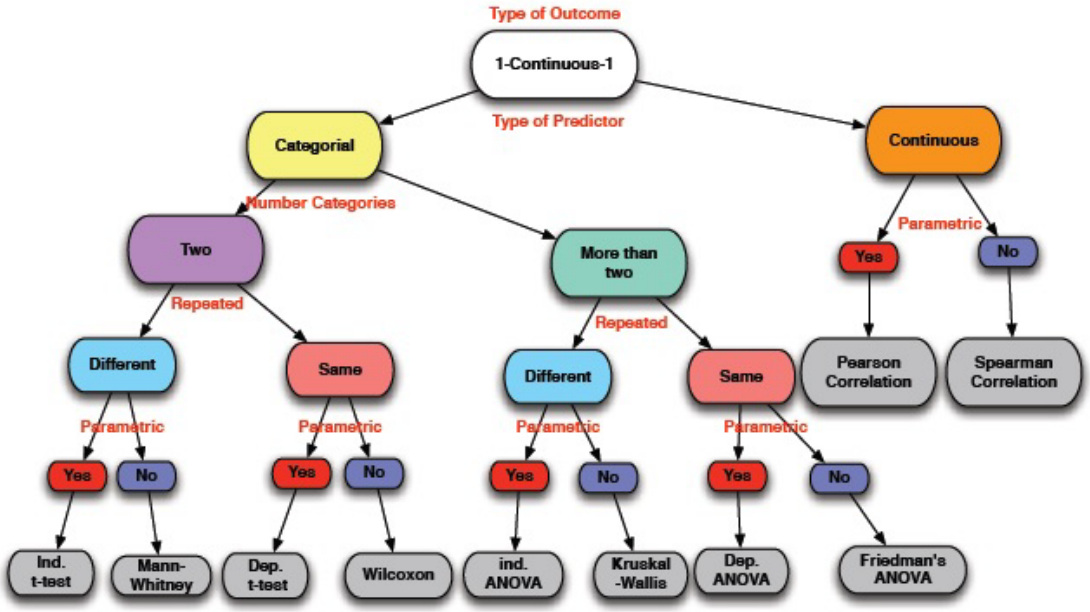

Figura 28 – Árvore de decisão para teste estatístico a usar [33]**.**

Na Figura 28 pode-se observar uma árvore de decisão para escolha do teste estatístico de grupo a efectuar. Neste caso, para os dados em causa, o teste a usar é o "Friedman's ANOVA", pois temos dados categorizados, com mais do que dois conjuntos, com as mesmas condições iniciais em todos os testes e dados não paramétricos.

Tabela 4 – Estatística descritiva do conjunto de dados (melhores resultados) dos testes da primeira fase.

Estatística descritiva								
Teste	Nº execuções	Média	Desvio padrão	Min	Max	Percentis		
						25%	50% (Mediana)	75%
1	12	4,520461	1,7473294	2,2959	6,1588	2,626251	5,586805	6,043310
2	12	5,833137	1,0390563	2,6387	6,5424	5,819870	6,131912	6,305248
3	12	4,096131	1,5658245	2,6242	6,1003	2,628162	3,781381	5,675116
4	12	3,535388	1,4297460	2,6116	6,1684	2,613387	2,628585	4,592992

Tabela 5 – Teste Friedman do conjunto de dados (melhores resultados) dos testes da primeira fase.

Teste Friedman	
Nº execuções	12
Distribuição χ^2 (Chi-square)	11,700
Graus de liberdade (df)	3
Significância assintótica (Asymp. Sig.)	,008
Significância exacta (Exact Sig.)	,006
Ponto de probabilidade	,001

Após análise das duas tabelas anteriores (Tabela 4 e Tabela 5), começa a verificar-se que o teste dois é o que mais se destaca pela positiva, pois na primeira tabela observa-se que em todas as colunas apresenta melhores valores. A média é a mais elevada, o que em conjunto com um desvio padrão inferior aos restantes, posiciona o conjunto de dados dois num valor superior aos outros. Na segunda tabela, no teste de Friedman, através dos 3 graus de liberdade em conjunto com o valor de χ^2, confirma-se que existem diferenças significativas no grupo de testes, com um grau de confiança de 99% [33].

Em seguida apresenta-se o Boxplot do mesmo conjunto de dados, de forma a validar visualmente o referido anteriormente.

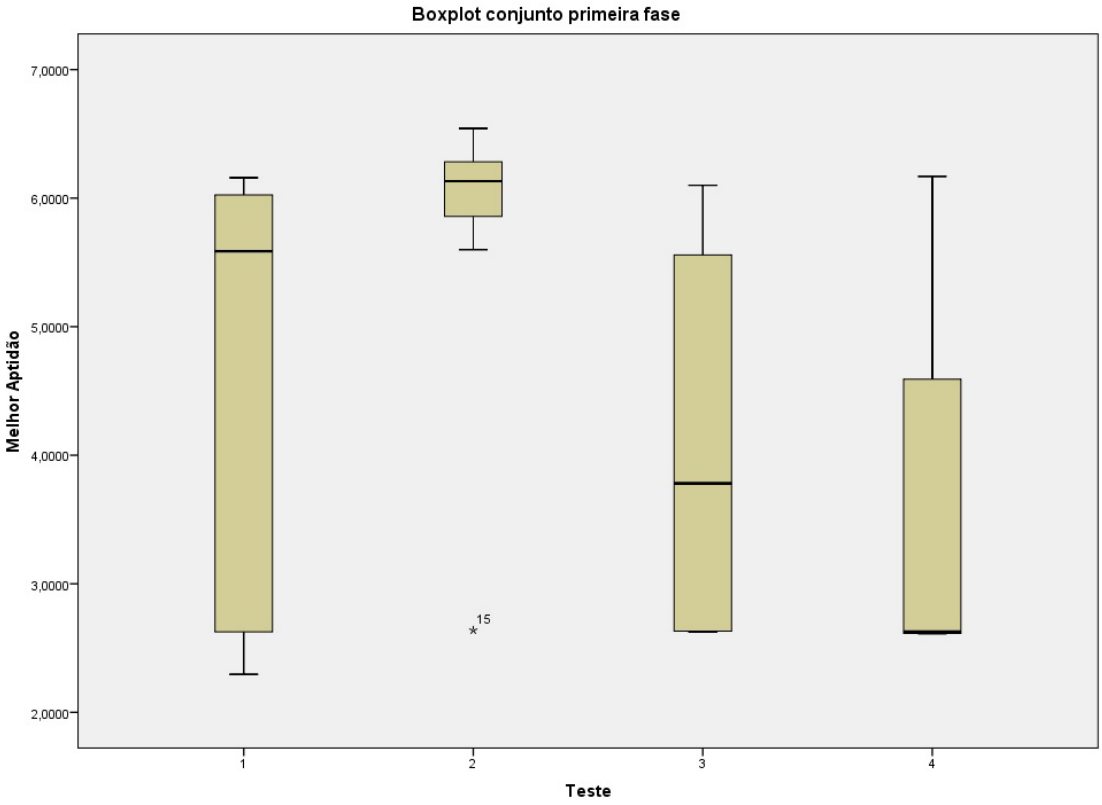

Figura 29 – Boxplot conjunto da melhor aptidão dos testes da primeira fase.

No Boxplot na Figura 29, verifica-se visualmente que o teste dois é o mais positivo de todos, tendo uma média na ordem dos 5,83 e apresenta uma menor dispersão dos dados.

4.1.3. Conclusão

Após a análise conjunta anterior, das tabelas e do Boxplot, podemos afirmar que dos quatro testes efectuados, o que apresentou melhores resultados foi o teste dois, sendo por isso o que reflecte os valores para a segunda fase dos testes. Todos os melhores resultados obtidos em cada um dos testes obtiveram estabilidade no modelo quando testados no sistema de física.

A título de curiosidade sobre a fase um dos testes, ficam algumas informações relativas aos dados obtidos (Tabela 6):

Tabela 6 – Informação variada sobre a primeira fase de testes (tempo execução, espaço em disco).

	Tempo máximo execução (mnts)	Tempo mínimo execução (mnts)	Tempo médio de execução (mnts)	Espaço ocupado em disco (GB)
Teste 1	2174,02	578,75	1312,10	
Teste 2	2443,96	589,09	1766,68	
Teste 3	2833,55	612,66	1287,50	
Teste 4	2721,53	592,49	1001,20	
Fase 1	2833,55	578,75	1341,87	18,5

4.2. Fase 2

Com a segunda fase de testes pretendeu-se testar o sistema criativo implementado, de forma a se perceber se e quão criativo é o sistema. Os vários parâmetros alteráveis nesta fase passam pela comparação de imagens (activa com Google Imagens ou inactiva) e pelos testes de física ao modelo (activo ou inactivo). Os restantes parâmetros foram mantidos constantes de forma a se poder retirar algumas conclusões, tendo sido os valores de probabilidade de cruzamento e mutação de 0,5 e 0,1 respectivamente.

Foram feitos os seguintes conjuntos de valores a testar:

- Teste 5 → Etapa de comparação de imagem = Google Imagens;
  Etapa de validação da física do sistema = activa;
- Teste 6 → Etapa de comparação de imagem = inactiva;
  Etapa de validação da física do sistema = activa;
- Teste 7 → Etapa de comparação de imagem = Google Imagens;
  Etapa de validação da física do sistema = inactiva.

Neste caso não se analisaram os resultados em conjunto, pois não são comparáveis entre si, devido a alguma das componentes de aptidão estar inactiva ou estarem as duas activas nos testes. Em vez disso, foram analisados os resultados de cada teste independentemente, de forma a se poderem retirar algumas conclusões.

4.2.1. Teste 5 – Fluxo completo

O teste número cinco é o teste mais importante, pois foi onde se mantiveram todas as componentes activas, de modo a testar todo o fluxo do sistema desenvolvido. Este teste consistiu em efectuar doze execuções completas, com as respectivas configurações. A maior dificuldade obtida neste teste foi a limitação de invocações do Google Imagens por minuto, o que levou a que os testes fossem feitos com menos cadência. No geral obteve-se um resultado aceitável e com valores de aptidão perto dos 7,9 valores.

4.2.2. Teste 6 – Apenas componente física activa

No teste seis foi utilizada apenas a componente de física activa, não sendo usada a componente de comparação de imagens. Neste caso o valor máximo de aptidão do modelo desce de 12,5 para 6,5, pois a aptidão das imagens vale 6 valores. Este teste serve para validar se o sistema consegue adaptar o modelo de forma autónoma, através do uso do algoritmo genético, e se consegue que este mantenha o equilíbrio, inclusivamente com vento e chuva.

4.2.3. Teste 7 – Apenas componente Google Imagens activa

O teste sete foi realizado apenas com a componente de comparação de imagens activa, tendo ficado a componente de física do sistema inactiva. Neste caso o sistema não deixa de ter a aptidão do modelo quanto à sua criatividade, pois continua a pesquisar por imagens relevantes no Google Imagens. Não estando a componente de física activa, não será possível saber se os modelos são estáveis ou não. O valor máximo de aptidão neste caso baixa de um total de 12,5 para 7, pois a aptidão da parte física do sistema vale 5,5 valores. Esta caso foi criado para validar se o sistema, apenas com esta componente activa, consegue gerar modelos Lego considerados criativos e validar se a componente de física interferia no resultado de criatividade.

4.2.4. Estatística da fase 2

Nesta secção, apresentam-se alguns dados relevantes sobre os resultados obtidos nos testes da segunda fase, disponíveis na Tabela 7.

Tabela 7 – Estatística descritiva do conjunto de dados (melhores resultados) dos testes da segunda fase.

Estatística descritiva							
	Nº execuções	Média	Desvio padrão	Min	Max	Max possível	Max/Max possível
Teste 5	12	7,90692874	1,4816367	4,7287502	8,99488735	12,5	0,71959099
Teste 6	12	3,98874998	1,4308932	1	4,86250019	6,5	0,74807695
Teste 7	2	5,80276942	0,5370038	5,4230504	6,18248844	7	0,88321263

Pode observar-se que foram obtidos resultados aceitáveis, pois em todos os testes se conseguiu um valor superior a 70% do valor máximo. Verifica-se ainda que o teste 7 foi o que obteve melhores valores, apesar de se terem realizado menos execuções, foi o que obteve um valor final mais elevado em conjunto com o desvio padrão mais baixo.

Nota: Não foi possível realizar todos os testes previstos para o teste 7 devido a limitações temporais.

4.2.5. Conclusão

Como referido anteriormente, este conjunto de testes não pode ser comparado directamente, pois a escala possível de resultados é diferente. Vamos analisar cada teste independentemente e efectuar uma pequena análise conjunta no final.

No teste cinco, o único comparável directamente com os da fase anterior, os resultados obtidos em termos de valores são melhores. Possivelmente por as imagens a comparar variarem, vindo directamente do Google Imagens. Conseguiu-se obter valores perto dos 7,9, o que significa que estão na ordem dos 71% do máximo, confirmando que é possível obter valores mais elevados.

No caso do sexto teste, onde apenas a componente de física esteve activa, conseguiu verificar-se o proposto. Em todas as execuções conseguiu-se gerar modelos que se foram adaptando, não tendo equilíbrio ao início, mas que através do algoritmo genético, foram modificados de forma a serem equilibrados no fim.

O sétimo teste, que utilizou apenas a componente de comparação de imagens com o Google Imagens activa, serviu para verificar se o sistema de física trazia alguma entropia ao sistema. Pelos resultados obtidos, verifica-se que enquanto o modelo não é equilibrado nas várias situações, a comparação de imagens não é muito eficaz. Após o modelo estar equilibrado, a comparação já consegue ser mais eficiente, provavelmente pela fórmula final de aptidão necessitar de alguns ajustes. Foi o teste que obteve melhores valores, ficando com um máximo na ordem dos 88% do possível. De forma a retirar mais conclusões sobre as diferenças entre este teste e o teste 5, seriam necessários mais testes conjuntos.

Como efectuado na fase um dos testes, ficam também algumas curiosidades sobre a segunda fase de testes (Tabela 8):

Tabela 8 – Informação variada relativa à segunda fase de testes (tempo execução, espaço em disco).

	Tempo máximo execução (mnts)	Tempo mínimo execução (mnts)	Tempo médio de execução (mnts)	Espaço ocupado em disco (GB)
Teste 5	6984,99	3433,11	4774,91	
Teste 6	1855,67	432,01	1183,61	
Teste 7	3727,46	3198,22	3462,84	
Fase 2	6984,99	432,01	3140,45	63

Confirma-se que o teste 6 é o mais rápido, pois tem a componente de imagens desactivada, não precisando de aceder à internet e descarregar imagens para o disco. Verifica-se também que o espaço ocupado em disco subiu consideravelmente, pois depende muito das imagens que o Google Imagens devolve.

No capítulo seguinte faz-se uma conclusão geral sobre todos os resultados obtidos em ambas as fases de testes.

5. Conclusões

Neste capítulo são analisados todos os resultados obtidos em ambas as fases dos testes e também o projecto de uma forma geral.

Este trabalho consistiu na implementação de uma aplicação que se pretende que seja criativa na criação de modelos Lego, efectuando para isso uma comparação das imagens dos modelos gerados com o Google Imagens. Foi também incluído uma componente de testes de física nos modelos (estabilidade com e sem chuva e vento), de modo a ficar mais completo e inovador. Para tal, foi usado um sistema de física para simulação de um ambiente real, onde para além de servir para testar as peças do modelo quanto à gravidade, também as sujeita a vento e chuva, como o podem ser na realidade. Todo a fase de comparação de imagens e testes de física compõem a parte de aptidão do sistema, que é usada pelo algoritmo genético para cálculo do fitness de cada modelo Lego. Todo o processo está encaixado dentro de um algoritmo genético, para que toda a população de modelos seja evoluída durante várias gerações de forma autónoma com o objectivo de melhorar os modelos criados inicialmente.

O sistema tem como início, a criação de uma população inicial de modelos de forma aleatória (tendo em conta algumas configurações), e o seu respectivo cálculo de fitness (comparação de imagens e física do sistema). Após esta fase entra-se no ciclo do algoritmo genético, onde são escolhidos os melhores modelos através da selecção dos pais implementada, que vai dar origem aos modelos a ser cruzados e mutados. Calcula-se o fitness da nova população gerada (modelos filhos) e escolhem-se os melhores através da selecção dos sobreviventes. Todo o processo é feito de forma autónoma, sem interacção do utilizador. Quando se obtém as condições de paragem de evolução da população (nº de gerações neste caso), o sistema devolve o melhor modelo, tendo em conta a função de fitness implementada (comparação de imagens e sistema de física).

Analisando cada uma das componentes, pode-se observar resultados positivos na área de algoritmos genéticos, onde foram testados alguns parâmetros para a mutação e cruzamento e a sua aplicação aos modelos Lego 3D. Na componente de física do sistema, conseguiu-se simular com alguma exactidão a gravidade, vento e chuva aplicada a modelos Lego 3D, algo que não foi encontrado na pesquisa efectuada. Na área do Lego, conseguiu-se gerar modelos Lego 3D e guardá-los em ficheiro ldr, o que permite a partilha e interacção nas várias aplicações da comunidade LDraw e Lego. Nas componentes do Picasa, validou-se que o sistema é fiável e robusto, pois funcionou correctamente no envio e disponibilização das mais

de setenta e duas mil imagens usadas. Em relação ao Google Imagens conseguiu-se efectuar uma obtenção de imagens de resultado para comparação nunca antes feita, onde se detectou que existe protecção por parte do Google Imagens para o número de consultas por minuto, mas foi superada limitando as consultas por minuto. O teste onde se obteve melhores resultados foi no teste 7, que ficou perto dos 88%, possivelmente por ter a componente de física inactiva. Verificou-se que o Google Imagens é uma poderosa aplicação que poderá ser usada no futuro para as mais variadas situações que necessitem de encontrar imagens semelhates.

De uma forma geral, a aplicação com todas as suas componentes funciona como um sistema criativo, podendo ou não obter modelos considerados criativos por quem os visualiza. Todas as componentes utilizadas ficaram com um enquadramento quanto à sua robustez e à sua utilização num nível mais profissional. Foi um trabalho bastante positivo e interessante de se fazer, que também serviu para aprofundar os conhecimentos nas várias áreas do projecto.

6. Futuro

Através da análise dos resultados conseguidos e da experiência obtida no desenvolvimento e execução do projecto, conseguiu-se identificar eventuais melhorias na abordagem efectuada. Existem muitos caminhos por onde seguir, seja pela área da criatividade, da genética ou do Lego. Vamos separar as melhorias propostas por essas três áreas, de forma a quem continuar este trabalho consiga identificar o caminho a seguir da forma mais correcta.

Criatividade

Esta é a área onde é mais difícil de identificar pontos concretos de evolução, pois a própria criatividade não tem uma definição aceite por todos e continua a gerar muita controvérsia. De modo a evoluir este trabalho nesta área, propõe-se que se implementem mais métodos de validação de quão criativo é o modelo gerado (neste caso usou-se o Google Imagens). Sugere-se o uso de comparações em todas as formas possíveis com objectos já existentes (imagens do modelo Lego, modelo Lego, ficheiro ldr gerado, comparações a nível de física) de forma a conseguir obter semelhanças com objectos já existentes. A sugestão anterior permite ter mais fontes para reconhecer uma criatividade-P gerada pelo sistema. Podemos também melhorar a comparação de imagens implementada (pixel a pixel e histogramas de cinzentos) para que a comparação pelo sistema seja mais eficaz e se consiga obter melhores resultados.

Genética

Este ponto passa por testar a melhor forma de evolução do sistema implementado. Pode-se identificar pequenos passos, tais como, alterar os parâmetros do algoritmo genético (tamanho da população, mais probabilidades de cruzamento e mutação, aumentar as gerações, tamanho da elite ou o tamanho do torneio) e efectuar mais testes de forma a verificar se os resultados melhoram. Também se pode alterar componentes do algoritmo genético, tais como, usar uma geração de população inicial não aleatória, substituir o processo de cruzamento utilizado por um cruzamento uniforme, PMX (cruzamento parcialmente combinado), CX (cruzamento cíclico), outro existente ou até tentar criar um novo cruzamento específico para este caso. A mutação utilizada, com o objectivo de manter diversidade na população pode também ser substituída por uma existente ou criar uma mutação especifica para o caso de modelos Lego. No caso da selecção dos sobreviventes, esta componente também pode ser substituída, existem algumas abordagens possíveis, como por exemplo uma onde se incluí os pais para além dos filhos. Por fim, talvez a componente que também se relaciona mais com outras áreas, é a função de fitness, que pode ser melhorada e aumentada a sua eficácia, através de mais parâmetros que contribuam para o seu valor final.

Lego

No caso do Lego, surgem também várias abordagens possíveis, como por exemplo a inclusão de mais peças no sistema, tornando este mais completo em termos de peças Lego. Caso se introduzam peças que não sejam do tipo tijolo (rectangular), tem de ser alterado o sistema de colisões implementado e a própria criação do modelo no sistema de física. Podem ser introduzidos também outros tipos de encaixe, de forma a tornar o sistema mais complexo. Pode ser implementada uma interface com o utilizador, de modo a que este tivesse alguma interacção com o sistema. Esta alteração teria que ser feita de uma forma cuidada, pois caso o utilizador consiga encaminhar o modelo para um certo tipo, poder-se-ia perder a parte de criatividade do modelo. Para a componente do sistema de física, pode-se aumentar a realidade da simulação de condições do mundo real, através da implementação de outras validações e/ou detalhar melhor as já implementadas.

Como visto neste capítulo, muitos caminhos podem ser explorados tendo como base este projecto de mestrado, onde se tenta juntar a criatividade computacional, o Lego e os algoritmos genéticos.

7. Referências

[1] R. C. Schank and D. A. Foster, "The engineering of creativity : a review of Boden ' s The Creative Mind *," *Artificial Intelligence*, vol. 79, pp. 129–143, 1995.

[2] M. A. Boden, *The Creative Mind: Myths and Mechanisms*, Second Edi. Routledge, 2004, pp. 1–344.

[3] G. a. Wiggins, "A preliminary framework for description, analysis and comparison of creative systems," *Knowledge-Based Systems*, vol. 19, no. 7, pp. 449–458, Nov. 2006.

[4] G. Wiggins, "Searching for computational creativity," Goldsmiths' College, University of London, New Cross, London, 2006.

[5] K. F. Man, K. S. Tang, and S. Kwong, "Genetic Algorithms : Concepts and Applications," *IEEE Transactions on industrial electronics*, vol. 43, no. 5, pp. 519 – 534, 1996.

[6] G. L. Soares, "Algoritmos Genéticos : Estudo , Novas Técnicas e Aplicações," Escola de Engenharia da Universidade Federal de Minas Gerais - Centro de Pesquisa e Desenvolvimento em engenharia Eléctrica, 1997.

[7] G. Ritchie, "The JAPE riddle generator: technical specification," School of Informatics, University of Edinburgh, 2003.

[8] D. R. Hofstadter, "The COPYCAT project: An Experiment in Nondeterminism and Creative Analogies," Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1984.

[9] D. Cope, "EMI - Experiments in Musical Intelligence." [Online]. Available: http://artsites.ucsc.edu/faculty/cope/experiments.htm. [Accessed: 15-Jan-2013].

[10] P. Funes, L. Lapat, and J. B. Pollack, "EvoCAD: EVOLUTION-ASSISTED DESIGN." [Online]. Available: http://demo.cs.brandeis.edu/pr/buildable/evocad/aid00/. [Accessed: 04-Nov-2012].

[11] M. Peysakhov, V. Galinskaya, and W. C. Regli, "Representation and evolution of lego-based assemblies," Korman Computing Center, Drexel University, 2006.

[12] P. Petrovic, "Solving LEGO brick layout problem using Evolutionary Algorithms," EVAL - Department of Computer and Information Science, Norwegian University of Science and Technology, 2001.

[13] T. Santos, A. Ferreira, and M. J. Fonseca, "Using Sketches and Retrieval to Create LEGO Models," Department of Computer Science and Engineering, INESC-ID/IST/Technical University of Lisbon, 2008.

[14] LDraw Organization, "LDraw.org - Centralized LDraw Resources," 2012. [Online]. Available: http://www.ldraw.org/. [Accessed: 20-Oct-2012].

[15] Google, "Picasa," 2013. [Online]. Available: www.picasa.com. [Accessed: 14-Jan-2013].

[16] P. Machado, J. Romero, A. Santos, A. Cardoso, and A. Pazos, "On the development of evolutionary artificial artists," *Computers & Graphics*, vol. 31, no. 6, pp. 818–826, Dec. 2007.

[17] Autodesk Inc, "Autodesk Maya," 2013. [Online]. Available: www.autodesk.pt/maya. [Accessed: 15-Jan-2013].

[18] P. Machado and A. Cardoso, "All the Truth About NEvAr," *Applied Intelligence*, vol. 16, pp. 101–118, 2002.

[19] T. Collins and C. Coulon, "FreshJam : Suggesting Continuations of Melodic Fragments in a Specific Style," *Papers from the 2012 AIIDE Workshop*, pp. 73–75, 2012.

[20] D. Leake, "Creativity by Case-Based Reasoning (CBR): SWALE Project Home Page." [Online]. Available: http://www.cs.indiana.edu/~leake/projects/swale/. [Accessed: 03-Nov-2012].

[21] The LEGO Group, "LEGO," 2012. [Online]. Available: www.lego.com. [Accessed: 20-Oct-2012].

[22] P. J. Funes and J. B. Pollack, "Computer Evolution of Buildable Objects for Evolutionary Design by Computers," 1997.

[23] K. Sastry, D. Goldberg, and G. Kendall, "Genetic Algorithms," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Springer, 2005, pp. 97–125.

[24] G. M. Ribeiro and L. A. N. Lorena, "Roteamento de veículos dinâmico usando algoritmos genéticos," *XIX ANPET - Congresso de Pesquisa e Ensino em Transportes*, pp. 1–12, Jan-2005.

[25] F. L. Rodrigues, H. G. Leite, H. do N. Santos, A. L. Souza, and G. F. da Silva, "Metaheurística algoritmo genético para solução de problemas de planejamento florestal com restrições de integridade," *Revista Árvore*, vol. 28, no. 2, pp. 233–245, 2004.

[26] B. R. Donald, "A search algorithm for motion planning with six degrees of freedom," *Artificial Intelligence*, vol. 31, pp. 295–353, 1987.

[27] Y. Meng and B. Tiddeman, "Implementing the Scale Invariant Feature Transform (SIFT) Method," Department of Computer Science, University of St. Andrews.

[28] G. Pass and R. Zabih, "Comparing images using joint histograms," Computer Science Department, Cornell University, Ithaca, NY, 1999.

[29] S. Wang, "Applications of Fourier Transform to Imaging Analysis," University of Wisconsin-Madison, 2007.

[30] D. Calzolari, S. Bruschi, L. Coquin, J. Schofield, J. D. Feala, J. C. Reed, A. D. McCulloch, and G. Paternostro, "Search algorithms as a framework for the optimization of drug combinations.," *PLoS computational biology*, vol. 4, no. 12, pp. 1–14, Dec. 2008.

[31] A. Alexandrescu and I. Agavriloaei, "Determining the best mutation probabilities of a genetic algorithm for mapping tasks," Universitatea Tehnică „Gheorghe Asachi" din Iasi, BULETINUL INSTITUTULUI POLITEHNIC DIN IASI, 2011.

[32] N. M. Razali, Y. B. Wah, and M. Sciences, "Power comparisons of Shapiro-Wilk , Kolmogorov-Smirnov , Lilliefors and Anderson-Darling tests," *Journal of Statistical Modeling and Analytics*, vol. 2, no. 1, pp. 21–33, 2011.

[33] E. Costa, "Mestrado em Eng. Informática - Computação Evolucionária. Acetatos n^{o}6," Universidade de Coimbra - Faculdade de Ciências e Tecnologia, Coimbra, 2012.