LSystem2Unity - Desenvolvimento de um asset para aplicação de L-Sistemas na Unity

Pedro Victor da Silva Ferreira
line 1 (of Affiliation): dept. name of organization
UEA
Manaus, Brasil
pedro.victor.ferreira@outlook.com.br

Adriano Mendes Gil
line 1 (of Affiliation): dept. name of organization
UEA
Manaus, Brasil
adrianomendes.gil@gmail.com

***Resumo*—L-System é uma das diversas técnicas para gerar conteúdos de forma procedural em jogos digitais, baseada em sistemas de reescritas de regras, foi proposta inicialmente para o estudo de plantas, mas que demonstra uma boa capacidade para gerar outros tipos de estruturas. O artigo descreve o desenvolvimento de um plugin chamado de LSystem2Unity, que permite a aplicação de L-System em projetos no ambiente Unity, fornecendo uma interface amigável e com recursos para manipular regras de reescritas diversas vezes. Testes realizados apontam um bom desempenho do plugin desenvolvido**

***Keywords*-component; L-System; Jogos Digitais; Geração Procedural de Conteúdo, Unity;**

I. Introdução

Com a facilidade no desenvolvimento de jogos digitais criada nos últimos anos e com um mercado que ganhou bastante valor, tornou-se comum o surgimento de estúdios independentes, normalmente com equipes pequenas com poucos recursos e habilidades diversificadas em seus times, para poderem entregar seus projetos com conteúdos diversificados em pouco tempo a maioria demostrou interesse por técnicas de geração procedural de conteúdo (PCG).

O Sistema de Lindenmayer ou L-System[1] se tornou umas das técnicas bastante popular hoje em dia, no qual foi originalmente proposto por Aristid Lindenmayer para a modelagem de plantas, mas sua formula é capaz de ser utilizada em diversas aplicações, como a criação de fractais, geração de musicas em jogos digitais [2] , modelagem de plantas para jogos, criação de labirintos, etc. Graças a essa diversificação, o sistema acaba sendo bastante atraente para produzir essas estruturas através de gramaticas para reescrita.

O artigo trata-se do desenvolvimento de uma ferramenta para a engine Unity, com o objetivo de agilizar o desenvolvimento em jogos que utilizam o l-sistema, com isso sendo capaz de fácil aplicação para que desenvolvedores utilizem esse sistema de reescrita de estruturas em seus projetos, sendo capazes de reaproveitar as mesmas regras de substituição em outras cenas do jogo ou até mesmo em outros projetos.

O artigo esta organizado da seguinte forma: Na seção 02 é apresentado alguns trabalhos que utilizam o l-sistema; a seção 03 apresenta os conceitos que se encaixam ao projeto; a seção 04 mostra o desenvolvimento da ferramenta e suas etapas de execução; a seção 05 demostra a ferramenta construída e suas interfaces; na seção 06 é apresentado exemplos construídos com a ferramenta mostrando suas regras de produção, a seção 07 mostra o que foi atingido com a ferramenta, por ultimo, a seção 08 discuti alguns pontos a serem construídos e desenvolvidos na ferramenta em interações futuras.

II. Trabalhos Relacionados

Em [3] o l-sistema foi utilizado para a geração de labirintos usando a variação OL-System que permite a obtenção de diferentes resultados para cada vez que um cenário for criado.

Em [4] traz a geração de cenas virtuais 3D em tempo de execução, com a adição de um física rudimentar com gravidade e a capacidade de detectar colisões. A aplicação da a possibilidade de configurar diversos modos de execução, com duas principais, o de reprodução e de design.

Já [5] demostra a criação do L-System View para a geração de fractais com o l-sistema, implementando um interpretador de texto, aonde vão os comandos permitindo o usuário de modelar os fractais a serem produzidos, um interpretador que gera um l-sistema correspondente ao fractal e um desenho do fractal modelado, por fim apresenta um painel gráfico que mostra a interpretação gráfica do fractal modelado.

III. Referencial Teórico

A. L-System

Originalmente como uma proposta matemática para a geração e simulação de plantas, foi apresentada por Lindenmayer[1] para a área de botânica. Trata-se de um sistema que permite a construção de objetos complexos através de uma estrutura simples, tudo seguindo regras de substituição determinadas anteriormente, isso torna uma formula de reescrita de palavras, aplicando de forma paralela e simultânea a substituição de todos os elementos da palavra dada.

O seu sistema mais simples é denominado de DOL-sistema, onde se possui um alfabeto, axioma e o conjunto de regras de tratamento. Essa estrutura é uma gramatica determinística e livre de contextos onde todas as regras são aplicadas simultaneamente no axioma. Se for considerado um a palavra com duas letras 'a' e 'b', cada uma delas teria uma regra de reescrita. Considerando 'A' como o estado inicial da estrutura, onde possui as regras de produção como 'A' -> 'AB' e 'B' -> 'A', irá ter em sua execução durante suas gerações como na figura 1 .

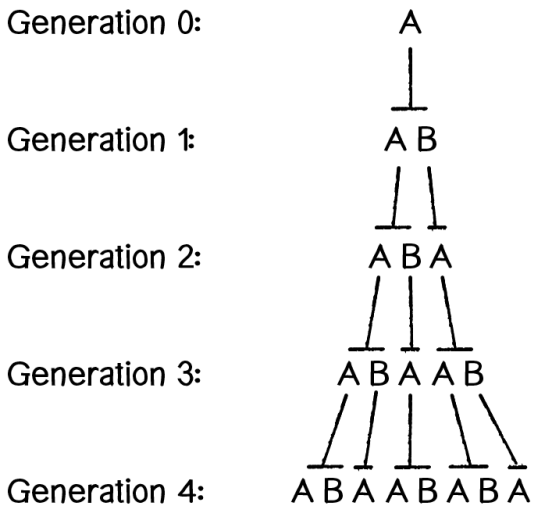

Figura 1. Gerações da estrutura seguindo a aplicação do L-System. Fonte: [6]

Pode-se criar diversas estruturas usando essa técnica, na figura 2 tem uma variação da curva de Hilbert 3D.

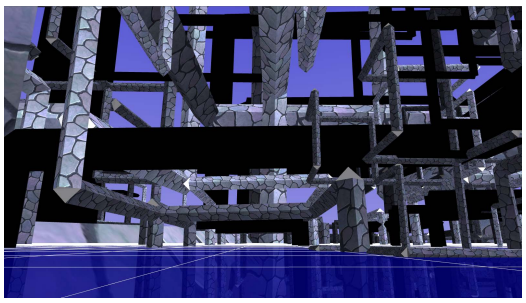

Figura 2. Exemplo de estrutura criada com L-System. Fonte: [4]

B. Turtle Graphics

Turtle Graphics[7] é o método da interpretação gráfica da estrutura criada pelo L-System, sua ideia básica é a suposição de uma tartaruga posicionada no plano cartesiano, a tartaruga é definida por uma tupla <(x,y), α >, onde as coordenadas (x,y) indicam a posição da tartaruga e α o seu angulo. Dando assim um passo d e um incremento de angulo fixo δ, a tartaruga ira poder responder a comandos simples como apresentados na tabela I

F	Mova um passo a frente de distancia d desenhando uma linha.
f	Mova um passo a frente sem desenhar uma linha.
+	Gira a esquerda por um ângulo.
-	Gira a direita por um ângulo.

Tabela I
EXEMPLOS DE SÍMBOLOS E COMANDOS

C. Geração Procedural

Em [8] refere a geração procedural de conteúdo á algoritmos capazes de gerar conteúdos para jogos, esses que podem ser dungeons, musicas, texturas, narrativas, entre outros.

Graças a esses algoritmos diminui os custos e tempo de produção desses elementos, dando a possibilidade aos desenvolvedores de se concentrarem em outros aspectos do jogo. Outra vantagem é que essas técnicas são capazes de aumentarem as variedades dos elementos no jogo, aumentando as experiencias do jogador durante suas jogatinas.

D. Unity 3D

Um motor de jogos proprietário desenvolvido pela Unity Technologies, permite a implementação dos jogos desenvolvidos nela em diversas plataformas como computadores (Windows, Linux, Mac), dispositivos moveis(Android,IOS), WebGL. Apesar da Unity ser um motor em C++, sua API(Application Programming Interface) para os desenvolvedores é em C# junto com a plataforma Mono, dando a capacidade de rodar em diversos dispositivos. Graças a essas funcionalidades e sua facilidade de desenvolvimento ela tornou-se um dos principais motores entre os desenvolvedores indies. Ela possui um serviço chamado Unity Asset Store, aonde os desenvolvedores buscam assets para contribuir em seus projetos, desde sprites de personagens e modelos 3D até sistemas de jogos completos.

IV. DESENVOLVIMENTO DA FERRAMENTA

Para o desenvolvimento da ferramenta foi observado a presença de duas etapas em torno do L-System, a primeira que é a parte de reescrita da estrutura inicial seguindo as regras de produção pre-definidas, a segunda etapa gira em torno da interpretação da estrutura final pelo algoritmo da tartaruga.

A segunda etapa por sua vez é mais abstrata, pois cabe ao desenvolvedor que for utilizar a ferramenta possa decidir como vai ser os passos da tartaruga para cada elemento dentro da estrutura final. Com isso foi criado duas classes MonoBehaviour: LSystem e TurtleAgent.

A. LSystem

Essa classe faz a implementação da primeira etapa, a de reescrever a estrutura por n gerações, nela o usuário irá poder definir a string inicial, números de gerações, tempo de espera entre as gerações da estrutura, se deve chamar os

Turtle Agents no final de cada geração e por ultimo mantem as regras de reescrita. Para o armazenamento das regras, foi criado um scriptableobject Ruleset, que armazena uma lista de objetos da classe Rule, por ser um scriptableobject pode se criar um único dicionario e direcionar ele para diversos LSystem que estão presentes na cena, não tendo a necessidade de repetir a mesma regra diversas vezes.

Após terminar a quantidade de gerações máximas definidas, a classe chama os objetos do tipo TurtleAgent passando a string final gerada, assim iniciando a segunda etapa do sistema que é de interpretação da estrutura.

B. TurtleAgent

A segunda etapa irá tratar da interpretação dos elementos presentes na estrutura final, para isso ele mantem uma serie de regras que possuem duas propriedades: Symbol e Action, um char e UnityEvent respectivamente. ScriptableObjects possuem certa limitação referente a objetos na cena, por não ter nenhuma ligação direta com a cena dentro da Unity, não é capaz de utilizá-lo para armazenar essa lista de interpretação.

Por utilizar o UnityEvent se cria possibilidade de passar as funções construídas e presentes nos GameObjects da cena, mantendo uma forma de deixar mais genérica a interpretação das estruturas.

V. Ferramenta Construída

A ferramenta desenvolvida (no qual está disponível no seguinte endereço https://github.com/iamPedroVictor/LSystem2Unity) possui um editor customizado que o torna mais amigável para o desenvolvedor, podendo construir suas estruturas com o L-Sistema, ela conta com:

- LSystem - Componente principal, aonde faz o processamento das estruturas com as regras de produção.
- TurtleAgent - Componente para leitura da estrutura fornecida pelo LSystem.
- Ruleset - Estrutura que armazena as regras de produção que vão ser utilizadas pelo LSystem.

A. Fluxo da ferramenta

Em seu atual momento a ferramenta começa seu processo assim que a função Start do MonoBehaviour é chamado, seu fluxo pode ser visto pelo diagrama na figura 3

B. Interface da ferramenta

O componente principal para a geração das estruturas esta no monobehaviour LSystem demostrado na figura 4, nele tem as propriedades principais, além do campo para por uma lista de regras de substituição.

A figura 5 apresenta a interface de Ruleset já customizada, aonde o usuário pode ver com mais facilidade os símbolos que vão ser trocados, além de botões para adicionar e remover novas regras.

A interface do componente TurtleAgent possui uma semelhança, mas para economizar espaço e deixar mais fácil a

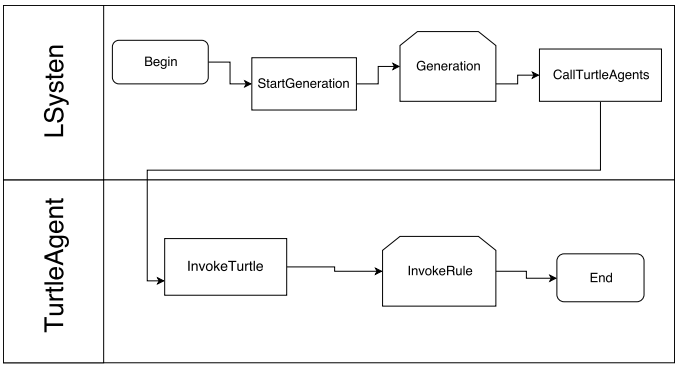

Figura 3. Fluxo da ferramenta LSystem2Unity

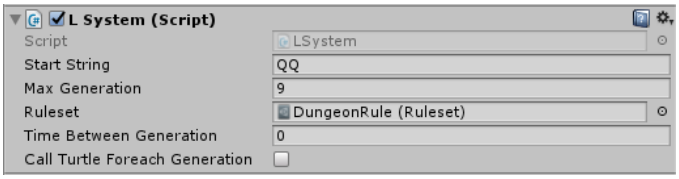

Figura 4. Interface do componente LSystem

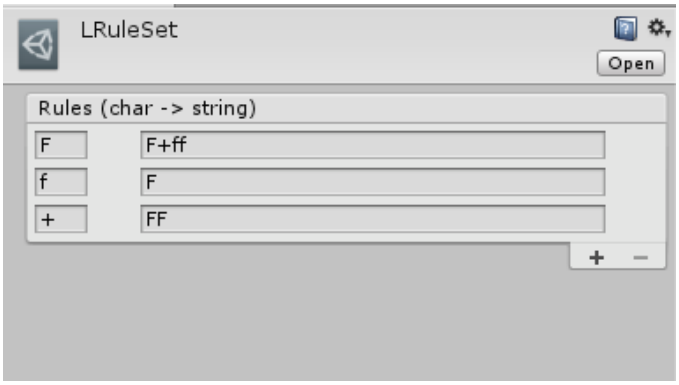

Figura 5. Interface de Ruleset

leitura, há uma funcionalidade de mostrar o conteúdo da lista somente quando o item for selecionado, isso é demostrado nas figuras 6 e 7, com o item não selecionado e selecionado respectivamente.

VI. Exemplos

Foi criado um gerador de cenários 2D para realizar os testes do plugin, o exemplo em questão foi rodado no editor do Unity 2017.1 em um computador com processador Intel Core i5-3470S de 2.90GHz, com 8Gb de ram. Durante o teste foi analisado o tempo de execução do processo durante as gerações. Foi feito um criador de cenários que monta um mapa distribuindo os pisos pelo cenário, é utilizado as seguintes regras da tabela II na geração do cenário.

Ao finalizar o TurtleAgent faz a leitura da estrutura gerada pelo LSystem e monta o mapa usando como referencia a tabela III. O cenário obtido com o axioma inicial de

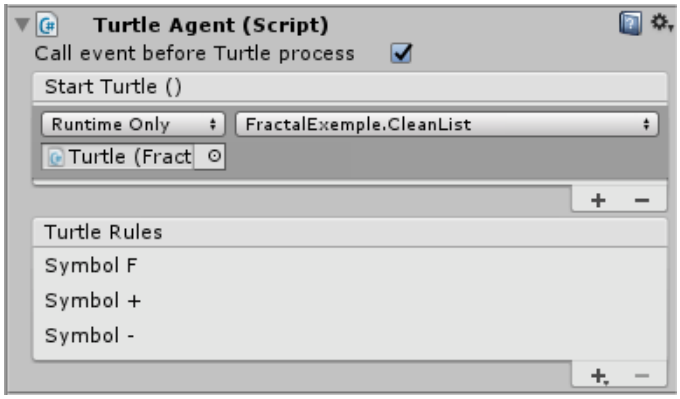

Figura 6. Interface de TurtleAgent sem item selecionado

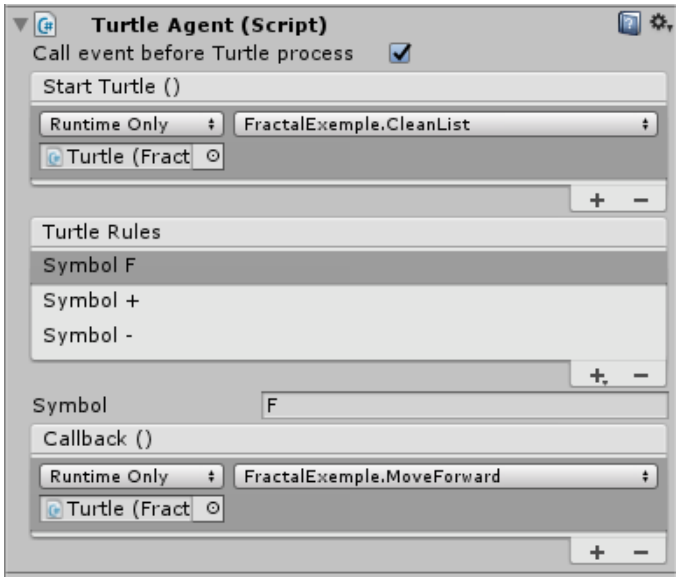

Figura 7. Interface de TurtleAgent com item selecionado

"QQ"com 8 gerações pode ser visto na figura 8 e na figura 9 apresenta um cenário gerado em 6 iterações com o axioma inicial "Q+".

É fácil a configuração e implementação das regras usando o editor, o sistema de reescrita é bom para geração de cenários, mas necessita de algoritmos mais especializados para geração mais complexas.

A. *Desempenho*

Utilizando as regras de reescrita da tabela II na seção VI, foi medido o tempo de execução por comprimento do

Q ⇒	QQ[Q
[⇒	+QQ]
] ⇒	QQ+QQ

Tabela II
CONJUNTO DE REGRAS DO L-SYSTEM

Q ⇒	Gera um quarto.
+ ⇒	Gira a direita a um ângulo de 90º.
- ⇒	Gira a esquerda a um ângulo de 90º.
[⇒	Salva a posição e rotação atual.
] ⇒	Volta para a ultima posição e rotação salva.

Tabela III
REGRAS DE INTERPRETAÇÃO DO TURTLEAGENT.

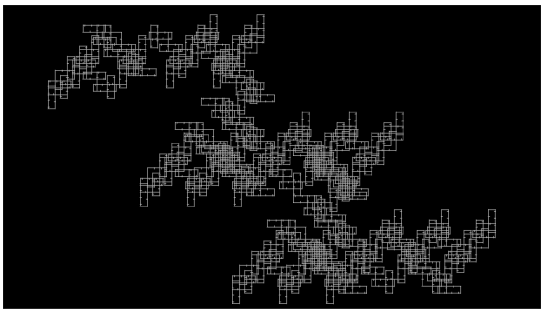

Figura 8. Cenário produzido na 8ª geração.

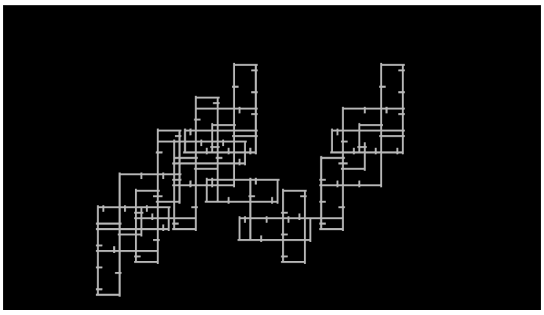

Figura 9. Cenário produzido na 6ª geração.

axioma, definindo 15 como o numero máximo de gerações os resultados estão presentes na tabela IV.

Comprimento do Axioma	Tempo(s)
32	0,0000142
124	0,0000293
472	0,0000917
25980	0,0046719
98792	0,0176053
1428512	0,2621754
20655832	03,7396957
78545636	14,1387742
298676752*	29,6070016

Tabela IV
TABELA COM O TAMANHO DO AXIOMA E O TEMPO PARA SER PROCESSADO EM SEGUNDOS. *GERAÇÃO INCOMPLETA

A ultima geração chegou a capacidade máxima suportada pelo stringbuilder, sua geração incompleta levou em torno de 34 segundos para ser realizada.

A figura 10 apresenta a tela do profiler na Unity sobre o uso da cpu durante os processos, aonde verifica-se a instabilidade de fps entre os processamentos, já na figura 11 tem

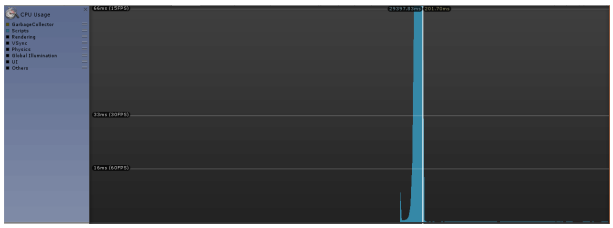

Figura 10. Profiler da Unity com o uso da cpu.

Figura 11. Profiler da Unity do uso de memoria.

o uso de memoria da ferramenta durante o processamento dos axiomas, chegando até 4,80 Gb alocados na memoria.

VII. Conclusão

No artigo foi descrito o desenvolvimento e a implementação do L-System no Unity atráves do asset denominado de LSystem2Unity, no qual dá aos desenvolvedores a possibilidade de aplicar o L-Sistemas em seus projetos com maior facilidade, desde criação de regras para o l-sistema, quanto ao interpretador TurtleAgent.

Durante sua aplicação, pode observar que ele é simples de utilizar e que consegue gerar estruturas rapidamente. Pode-se ver que pode criar fractais quanto cenários simples, a ferramenta conseguiu ter um processo eficiente durante as execuções das gerações. Ela apresenta limitações como somente um sistema de regras do L-System implementada, não dando assim, tanta opção de diversificação aos desenvolvedores, tendo assim que recorrer algoritmos mais complexos para ter mais variedades aos seus projetos.

A ferramenta consegue atingir seu objetivo de dar ao desenvolvedor a utilização do sistema de reescrita de estruturas, dependendo somente do desenvolvedor conecte seus algoritmos para construção de seus elementos usando o TurtleAgent.

VIII. Trabalhos Futuros

É planejado dar a ferramenta mais estruturas de regras, dando aos desenvolvedores uma maior variedade de opções nos desenvolvimentos de assets em seus jogos. Trazer mais funcionalidades ao editor do Unity, melhorando a utilização da ferramenta. Um ponto essencial é melhorar a performance, diminuindo os recursos de memória e demora de processamento das gerações em grande cadeias de caracteres.

A possibilidade da implementação para a produção de estruturas no editor, deixando os designers criarem conteúdos para os jogos, podendo optimizar e editar os elementos já produzidos.

Agradecimentos

Agradeço aos meus pais, Miguel Ângelo e Perenice Socorro, além a minha irmã Ana Virgínia por terem criado diversas oportunidades para mim e por sempre estarem me acompanhando na minha vivencia me apoiando, dando conselhos.

Ao meu professor e orientador Adriano Gil no qual me criou o interesse pelos campos de PCG e IA em jogos. Aos meus amigos da época de ensino médio e os que construir durante o curso, os quais me acompanharam durante esse anos, além dos professores Rodrigo Braga, Rafael Lima, Raphael Maquiné e André Machado.

A Vivian Fonseca por sempre me apoiar e a me incentivar no meu desenvolvimento profissional e acadêmico. A Universidade do Estado do Amazonas por ter proporcionado o curso Tecnólogo em Jogos Digitais e ao coordenador do curso Cleto Leal.

Referências

[1] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. New York, NY, USA: Springer-Verlag New York, Inc., 1990.

[2] M. Fridenfalk, "Algorithmic music composition for computer games based on l-system," in *2015 IEEE Games Entertainment Media Conference (GEM)*, Oct 2015, pp. 1–6.

[3] G. S. Etchebehere and M. A. Eliseo, "L-systems and procedural generation of virtual game maze sceneries."

[4] M. Fridenfalk, "Application for real-time generation of virtual 3d worlds based on l-system," in *Cyberworlds (CW), 2015 International Conference on*. IEEE, 2015, pp. 73–78.

[5] W. S. M. Santos[1] and S. do Lago Pereira, "Desenvolvimento de uma ferramenta para geração de fractais definidos por l-sistemas."

[6] D. Shiffman, "Nature of code, chapter 8. fractals," http://natureofcode.com/book/chapter-8-fractals/, [Online; Acessado em 18/03/2018].

[7] H. Abelson and A. A. DiSessa, *Turtle geometry: The computer as a medium for exploring mathematics*. MIT press, 1986.

[8] J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. O. Stanley, "Procedural content generation: Goals, challenges and actionable steps," in *Dagstuhl Follow-Ups*, vol. 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.