

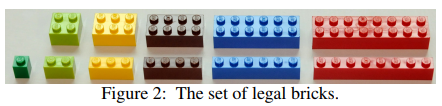
In figura de mai sus se observa o demonstratie a modelului nostrum de la inceput pana la sfarsit. Omuletul de LEGO este transformat in caramizi de dimensiuni 1x1, imbinate respectand culorile. Apoi structura este optimizata si caramizile inutile sunt indepartate. In final, instructiunile sunt produse si omuletul LEGO este produs din instructiuni.

**2. General pipeline**

Transformam reprezentarea in reatea a unui obiect intr-o multime de caramizi care nu sunt legate intre ele.

O caramida poate fi privita ca un pixel 3d. Alegerea rezolutiei unui obiect este importanta deoarece o valoare mai mare conduce la o aproximare mai buna a obiectului oridinal, dar creste numarul de caramizi necesare si timpul de executie.

Dupa ce convertim obiectul in caramizi de dimensiune 1x1, urmeaza sa imbinam caramizile in piese cat mai mari pana cand nu se mai pot face fuzionari(Ex in figura de mai jos)



În acest moment, modelul este foarte probabil să fie slab conectat și eventual deconectat conducând la o structura care nu poate fi construita. Prin urmare, crestem soliditatea modelul prin identificarea punctelor slabe și repararea acestora.

**2.1. Merge algorithm**

Pentru a reduce numărul de cărămizi și a crește conectivitatea, preferăm cărămizi mai mari. Putem face acest lucru prin simpla fuziune a caramizilor cu vecinii lor.

Folosim un “randomized greedy merge algorithm” care executa urmatorii pasi:

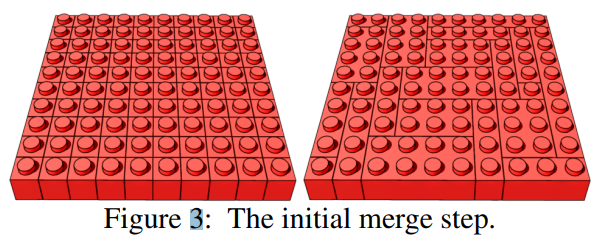
1. Alege o caramida din model la intamplare.
2. Cauta multimea de vecini valizi cu care poate fuziona
3. Fuzioneaza cu vecinul care are costul cel mai mic.
4. Sare la pasul doi pana cand nu mai exista vecini cu care poate fuziona
5. Sare la pasul 1 pana cand nu mai exista caramizi care pot fuziona

Pentru **pasul 2**, multimea de vecini valizi este cea din Figura 2.

La **pasul 3** favorizam caramida care creeaza cele mai multe conexiuni atunci cand fuzioneaza cu caramida curenta.

Daca doua fuzionari creeaza acelasi numar de componente, alegem la intamplare pe una dintre ele.

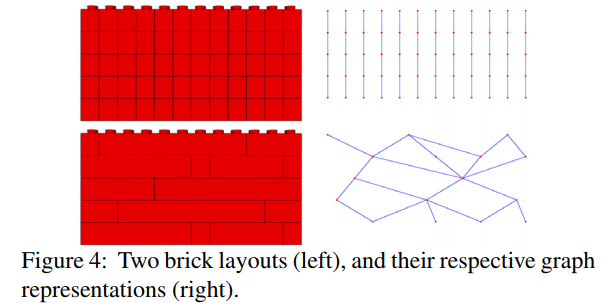
Rezultatul algoritmului poate fi vazut in figura de mai jos:

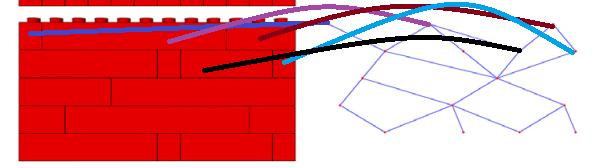


**2.2. Solidity Optimization**

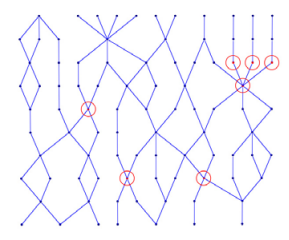
Stabilitatea modelului este legata de cum sunt conectate caramizile. Cu cat sunt conectate mai multe caramizi, cu atat modelul este mai solid.

Reprezentarea modelului poate fi facuta cu un graf, in care fiecare caramida reprezinta un varf si fiecare conexiune intre doua caramizi reprezinta o muchie. Ex in figura de mai jos:



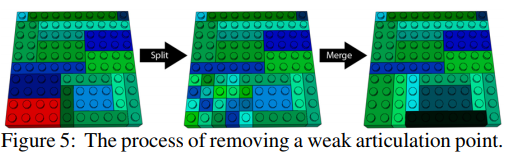


Doua subgrafuri care sunt conectate intre ele doar printr-o caramida, slabesc structura modelului.



Dupa aplicarea algoritmului de merge, caramizile sunt extinse la maxim. Prin urmare, impartimi fiecare caramida care fuzioneaza cu o alta caramida printr-un singur punct de articulatie, in caramizi de dimensiune 1x1, apoi aplicam din nou algoritmul de merge, schimband costul functiei cu unul random.

Aplicam acesti pasi pana cand numarul de componente conectate si numarul de puncte slabe de articulatie nu mai scad. In figura de mai jos putem observa procesul de splitare pentru elemental in rosu.



Experimentele arata ca sunt necesare mai putin de 50 de iteratii pentru a nu mai avea elemente deconectate sau puncte slabe de articulatie.

3. Extensions

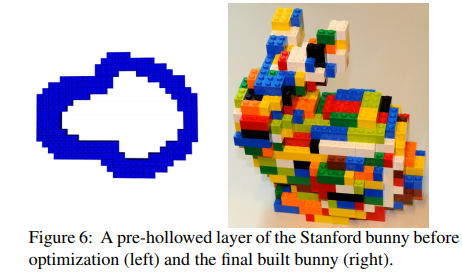
Pentru a reduce numarul de caramizi, se poate aplica o “pre-scobire” imediat dupa ce modelul a fost impartit in caramizi.

Dupa rularea algoritmului, atunci cand se considera ca modelul nu mai are puncte de articulatie slabe, se mai poate aplica o scobire.

**Reducing the overall brick number**

Putem renunta la caramizile din interiorul modelului, fara a afecta structura modelului. Exista doua strategii pentru scobirea modelului.

Pre-hollowing. Inainte de pipeline, utilizatorul poate specifica dimenziunea carcasei si poate elimina caramizile care depasesc numarul de caramizi de la fata exterioara a figurii.



Post-hollowing. O alta metoda este de a elimina caramizile din interior fara a modifica soliditatea modelului. Eliminam piesele fara a introduce mai multe elemente conectate sau mai multe puncte de articulatie slabe.

**Satisfying brick type limits**

Cand se construieste un model, exista un anumit numar de caramizi din fiecare tip. Pentru a respecta acest numar, folosim un algoritm asemanator cu post-hollowing. Eliminam caraminile peste limita, taindu-le in doua. Daca aceasta splitare provoaca puncte slabe, mergem la urmatoarea caramida.

**Using colors**. Putem folosi diferite colori pentru caramizi. In timpul pasului doi este necesara inca o verificare. Daca doua caramizi sunt vizibile si au culori diferite, ele nu pot fuziona.