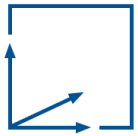
Simulating 2D Game Physics using dynamic Navigationgraphs

Lars Hinnerk Grevsmühl

16.04.2020

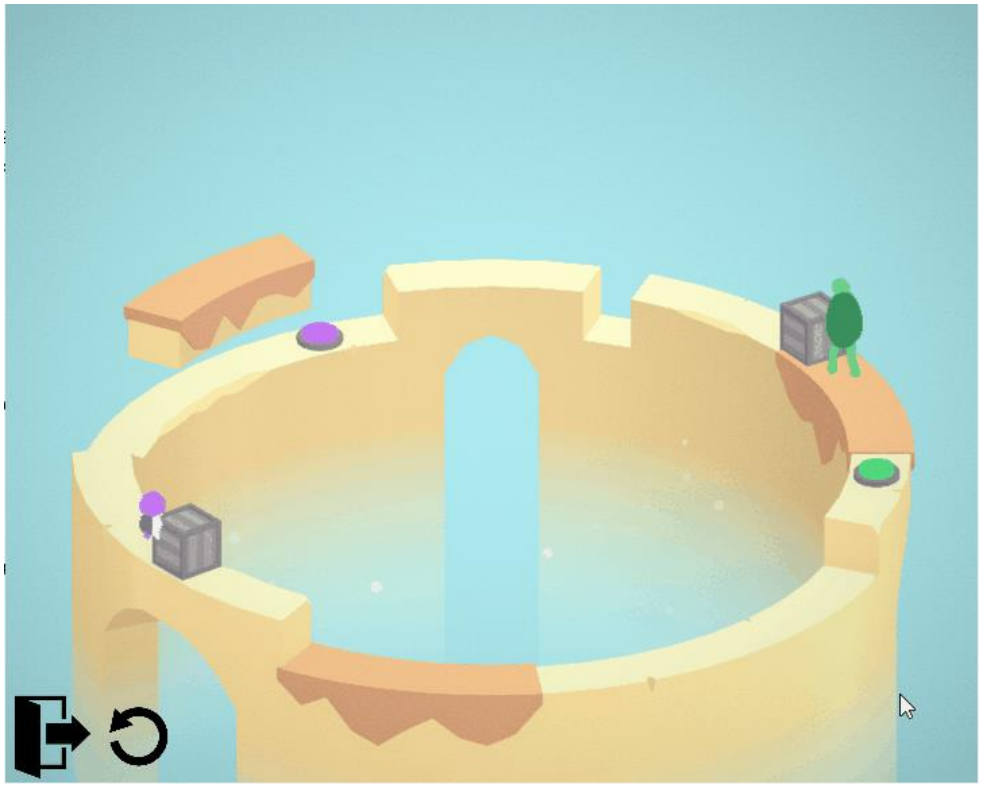


Final: Bachelor’s Thesis

Supervisor(s): Prof. Gudrun Johanna Klinker, Ph.D.

Introduction / Motivation

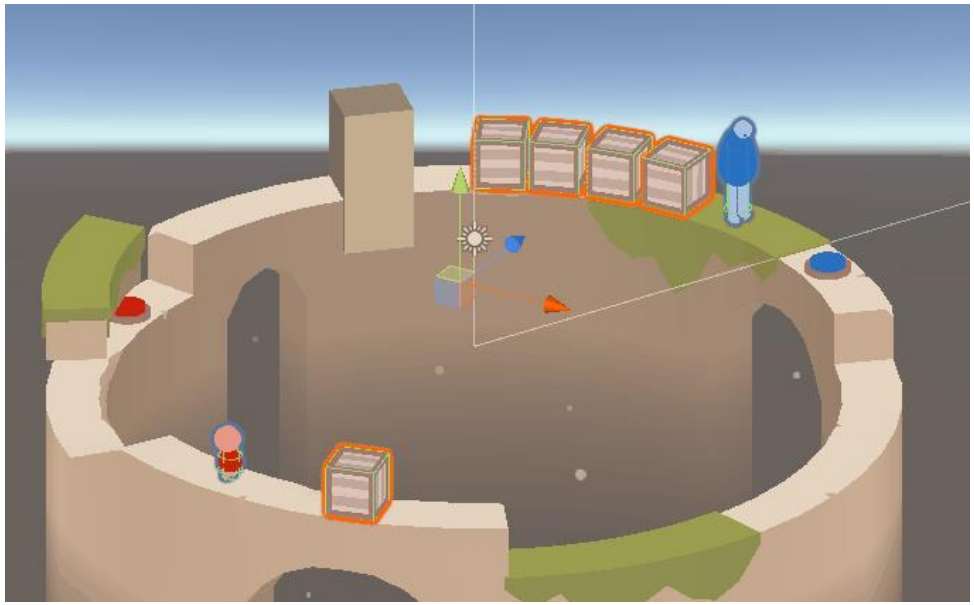
- Motivated by *„Huge and Cute“*
- puzzle game without jumping



Game Concept

Rigidbody Solutions

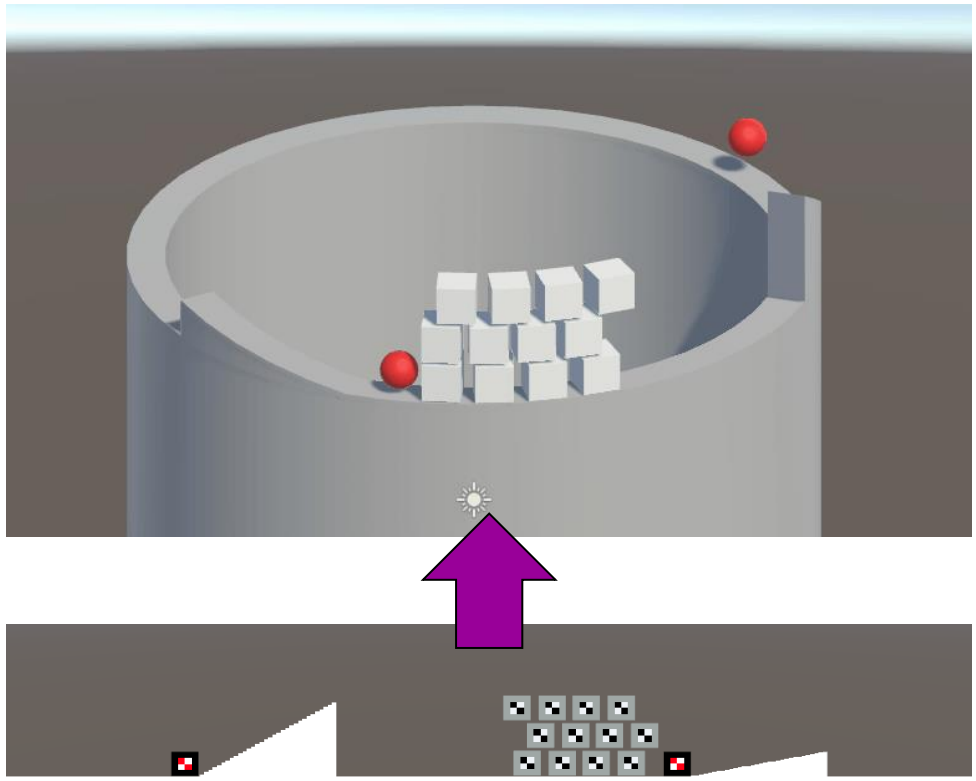
- **3D Rigidbodies** (stable, complicated, inefficient)



Game Concept

Rigidbody Solutions

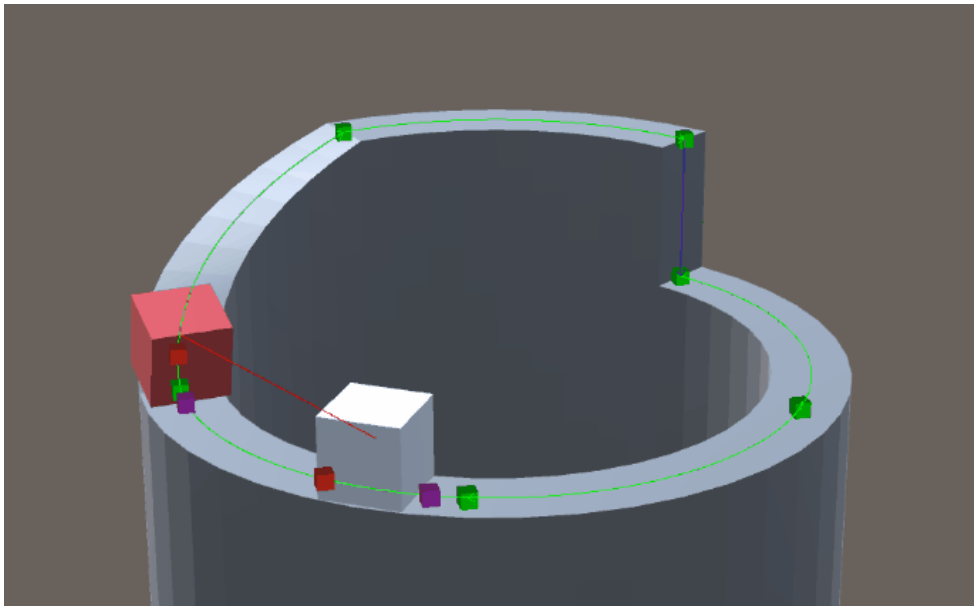
- **3D Rigidbodies** (stable, complicated, inefficient)
- **2D Rigidbodies** (unstable, complicated, efficient)
- **Goal:** Create a more stable and efficient solution



Game Concept

Proposed Graph based Solution

- Graphbodies instead of RBs
- Ground is represented by Graph
- Special Edges for Falling

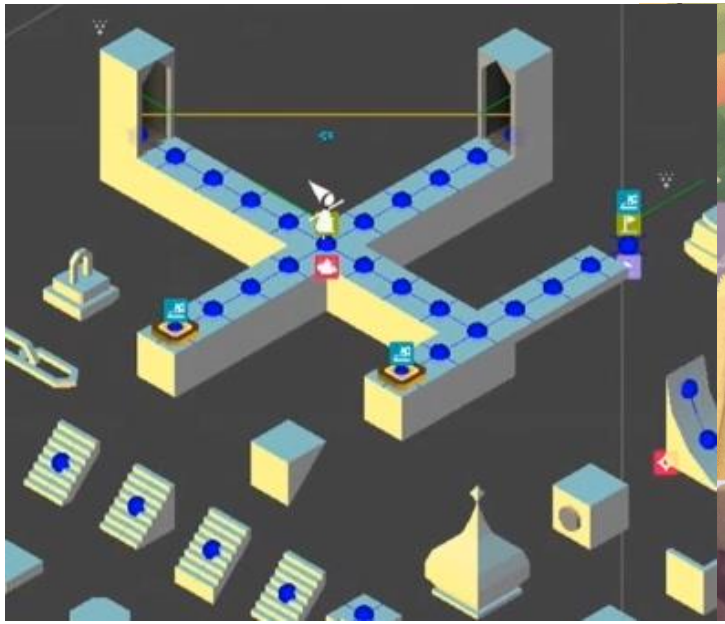


Prototype

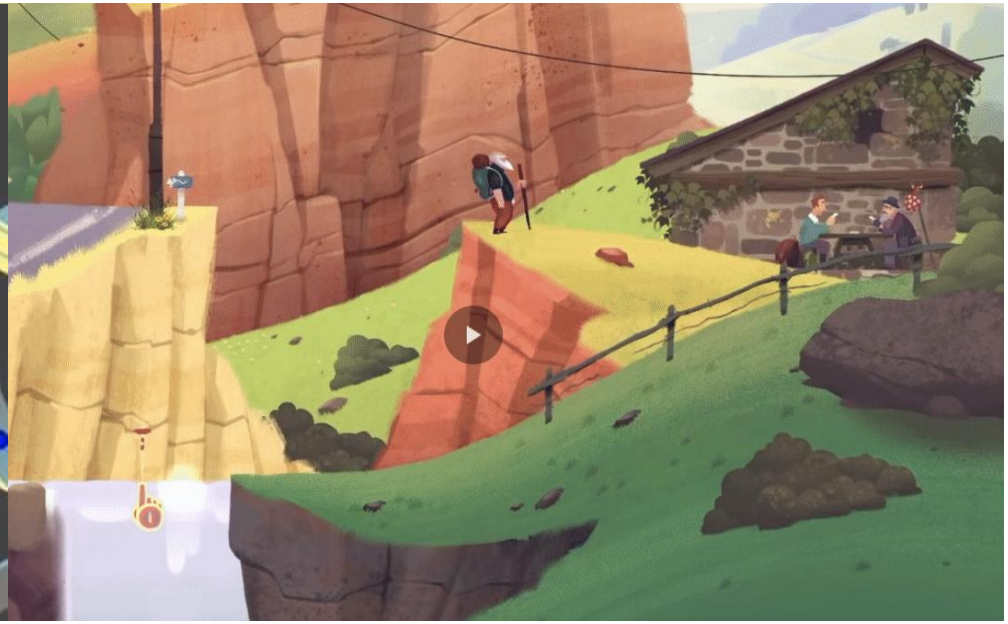
Existing Graph based Solutions

- No structured scientific research available

- Waypoint Systems: Monument Valley

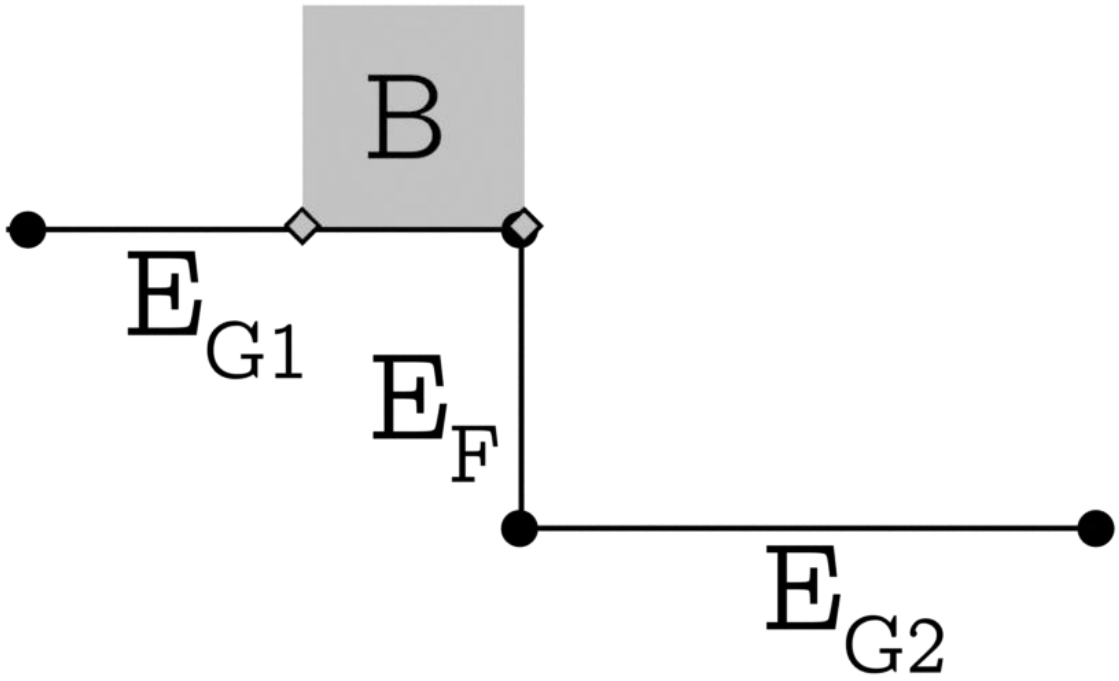


- Edge based Systems: Old Mans Journey



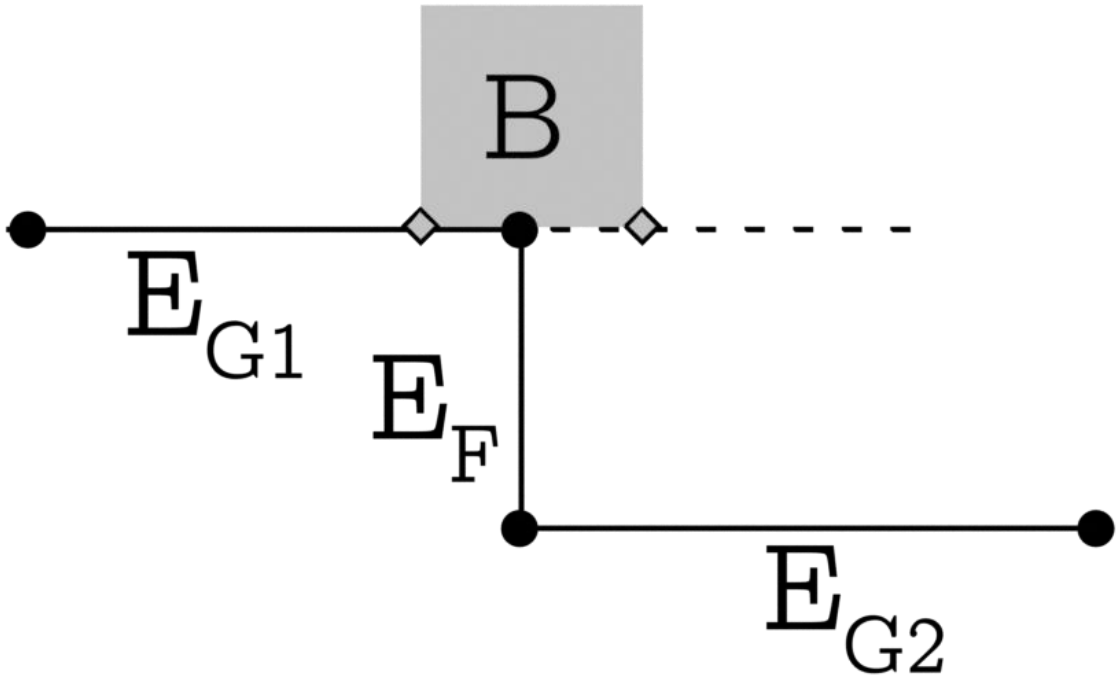
Implementation: Falling

- If Border reaches Vertex and Vertex has valid Falledge: go into overhang



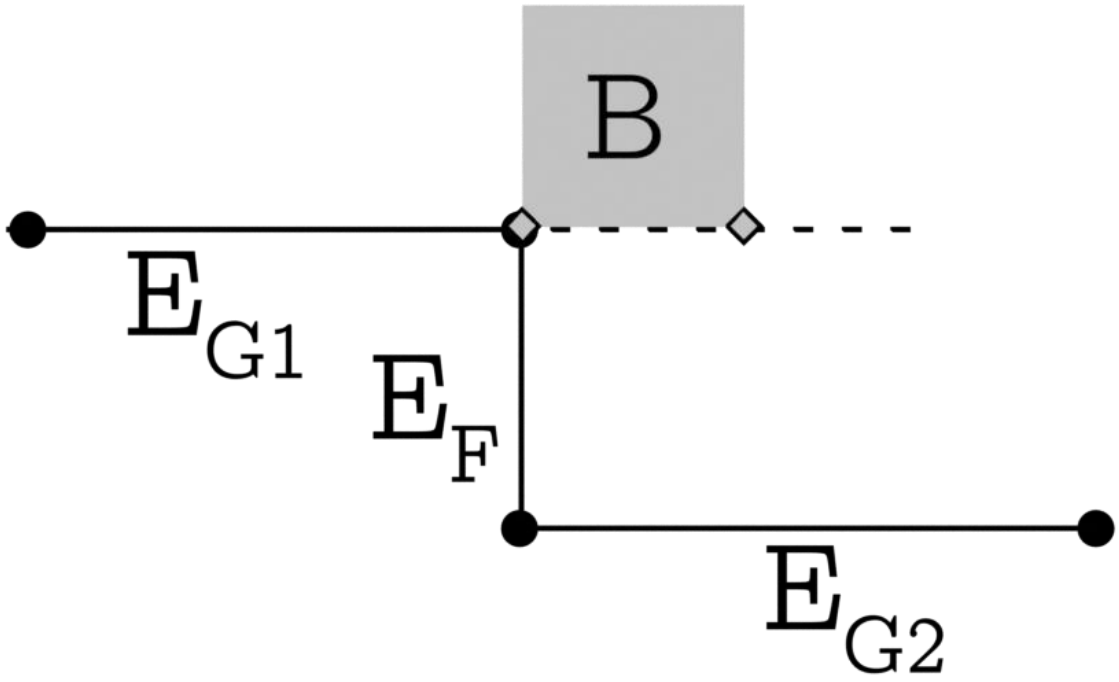
Implementation: Falling

- If Border reaches Vertex and Vertex has valid Falledge: go into overhang



Implementation: Falling

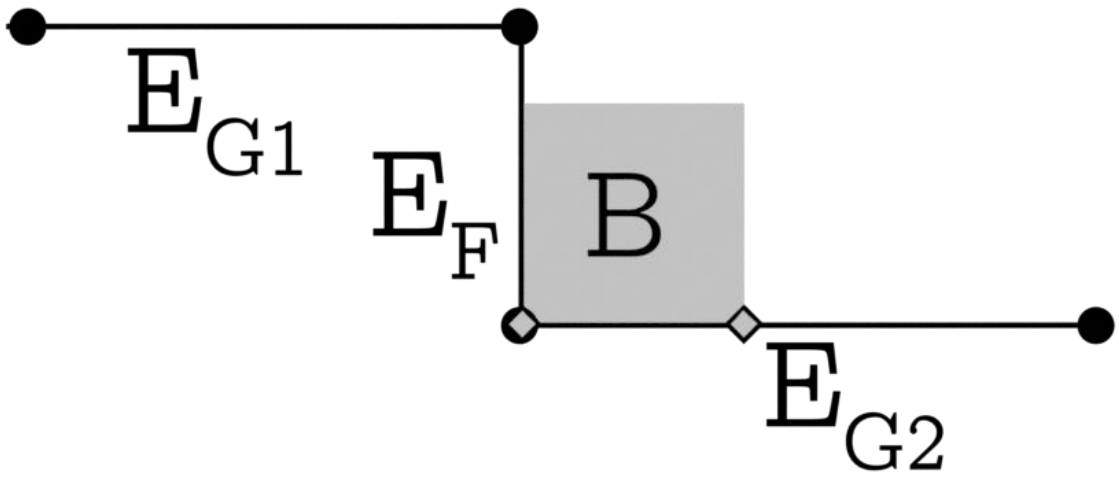
- If Border reaches Vertex and Vertex has valid Falledge: go into overhang
- If both in overhang: Fall to next Groundedge



Implementation: Falling

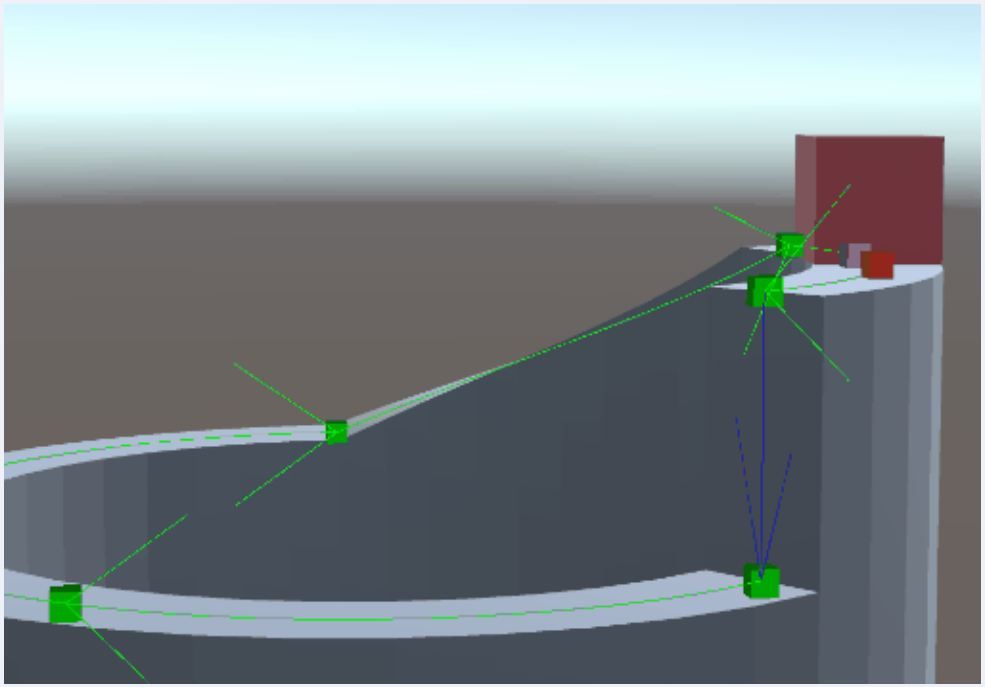
- If Border reaches Vertex and Vertex has valid Falledge: go into overhang

- If both in overhang: Fall to next Groundedge



Implementation: Falling

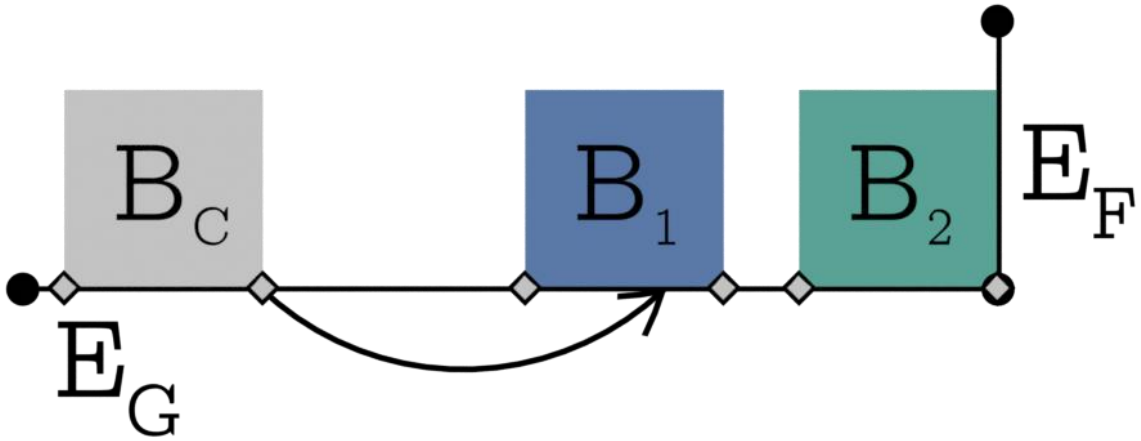
- If Border reaches Vertex and Vertex has valid Falledge: go into overhang
- If both in overhang: Fall to next Groundedge
- Visually Smoothed



Prototype

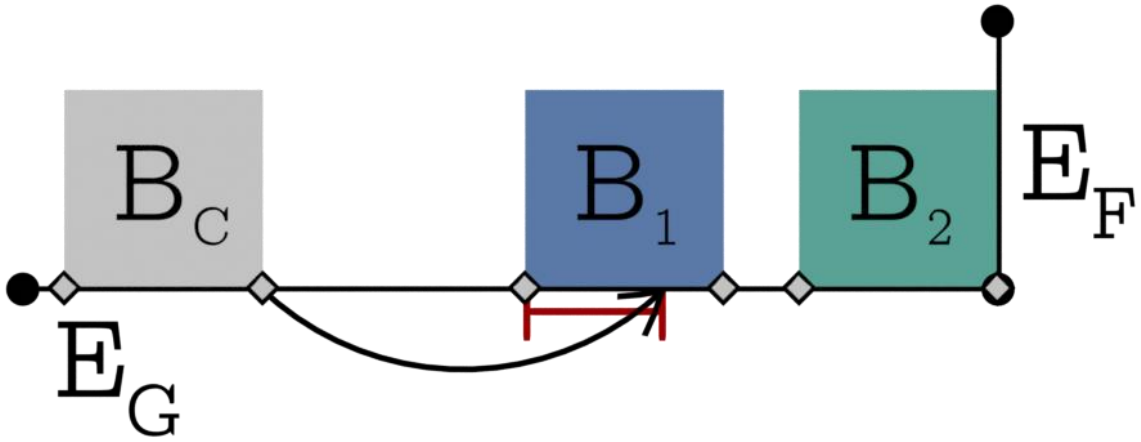
Implementation: Collision

- B_C tries to move for ist target_distance



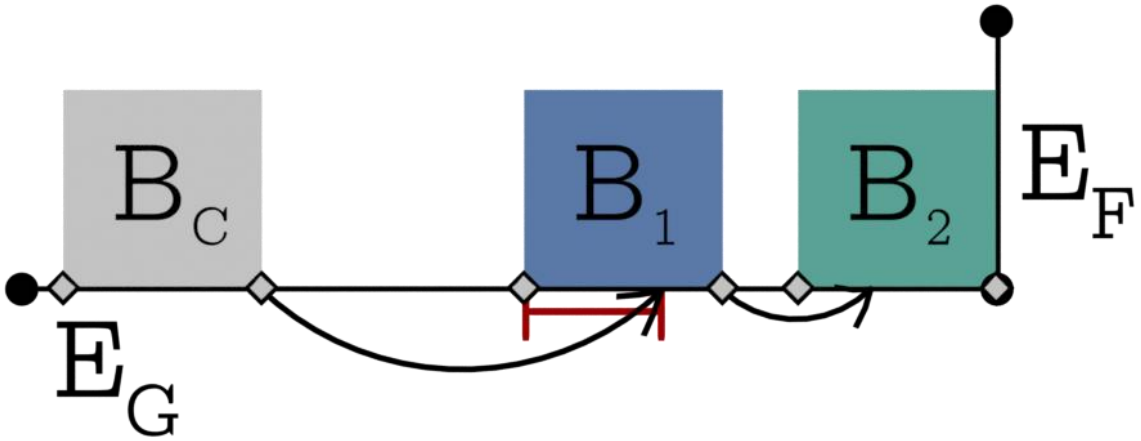
Implementation: Collision

- B_C detects collision with B_1 and calculates remaining_distance



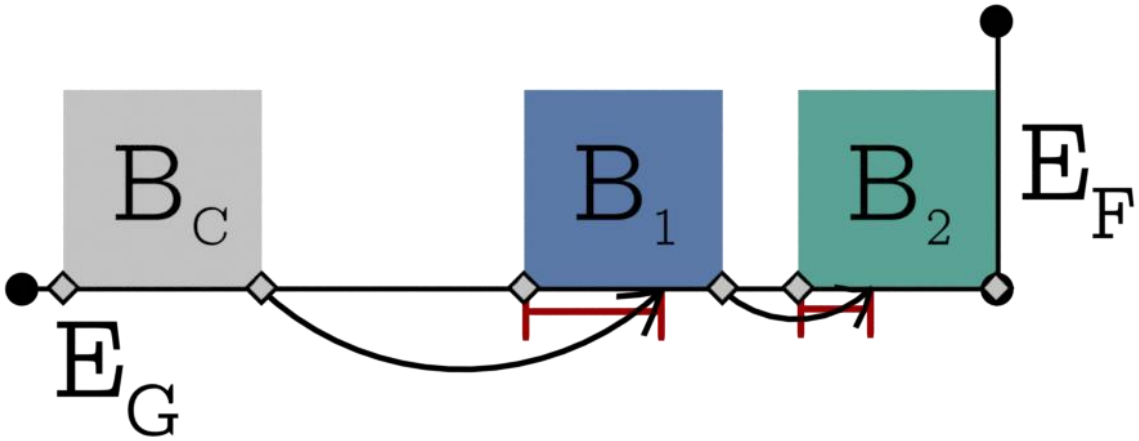
Implementation: Collision

- B_C sends movement request to B_1
- B_1 tries to move for its target_distance



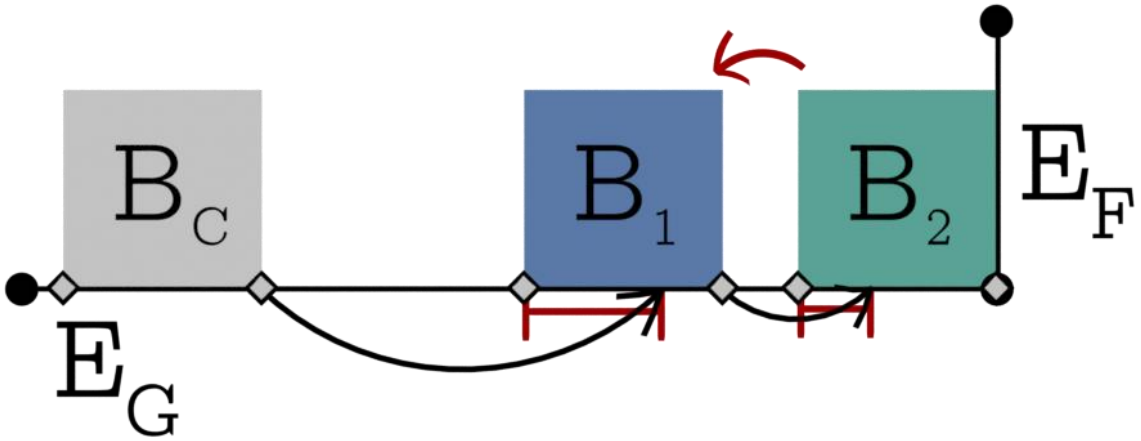
Implementation: Collision

- B_1 detects collision with B_2 and calculates remaining_distance



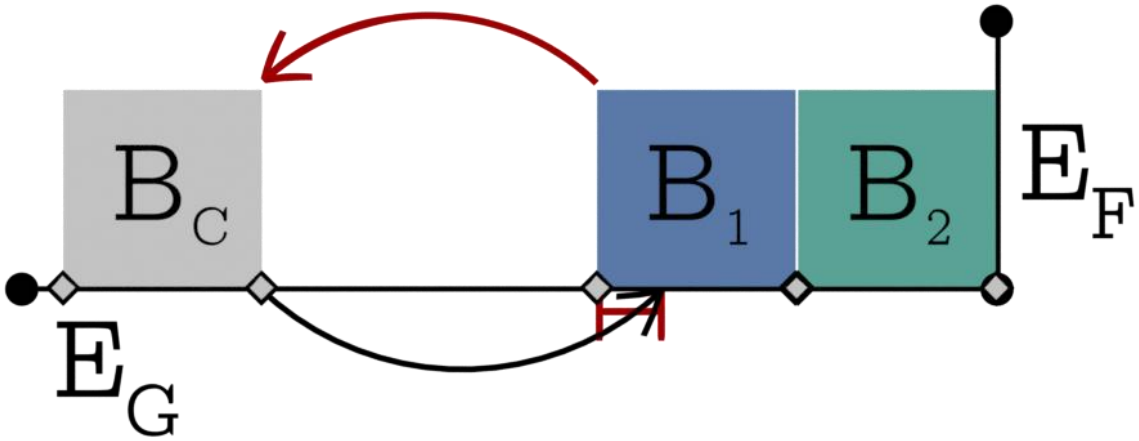
Implementation: Collision

- B_1 sends movement request to B_2
- B_2 tries to move for ist target_distance
- B_2 detects wall and reports back moved_distance



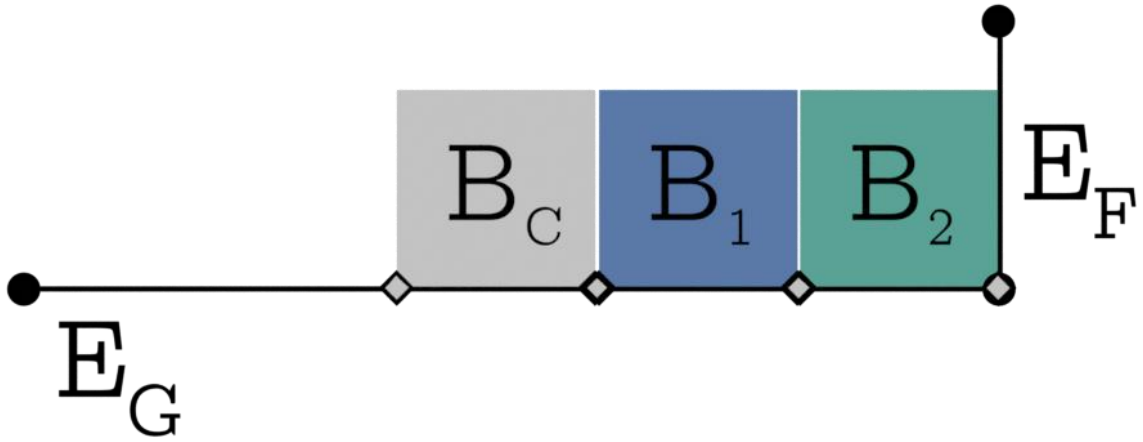
Implementation: Collision

- B_1 moves as far as possible
- B_1 reports back its moved_distance

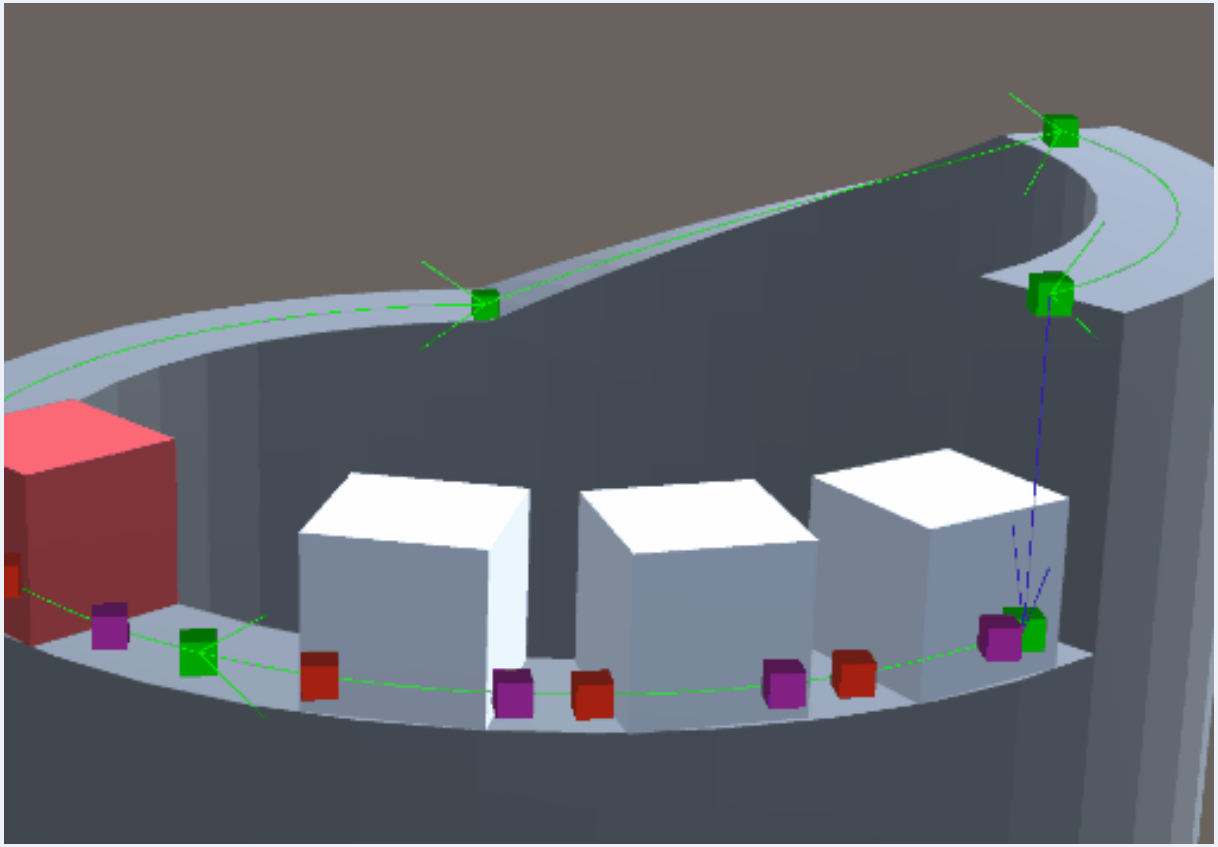


Implementation: Collision

- B_C moves as fasr as possible
- Chain of movement requests
- Very stable



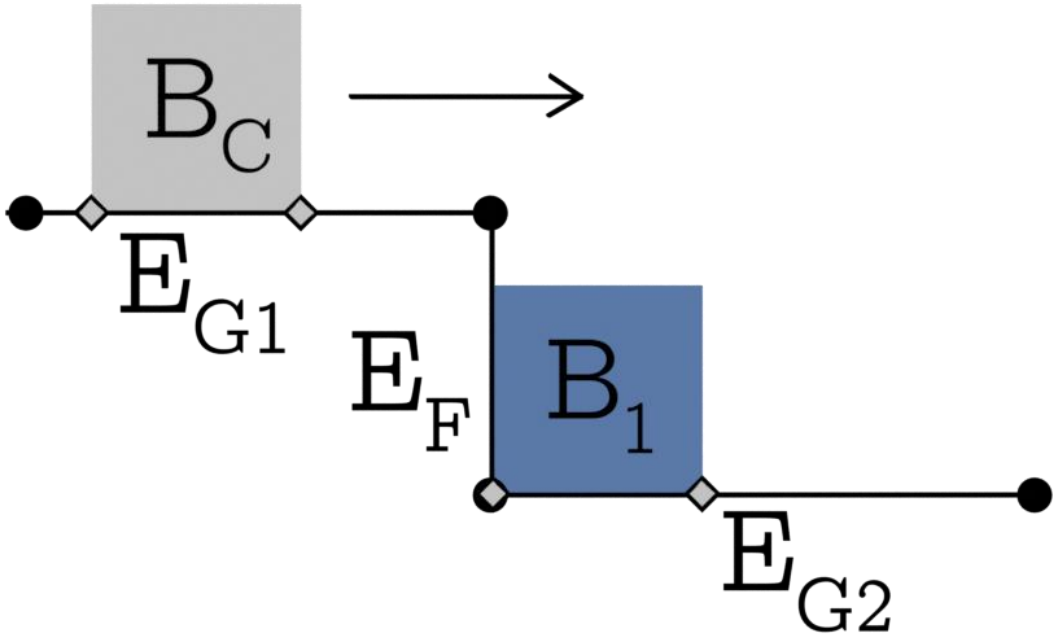
Implementation: Collision



Prototype

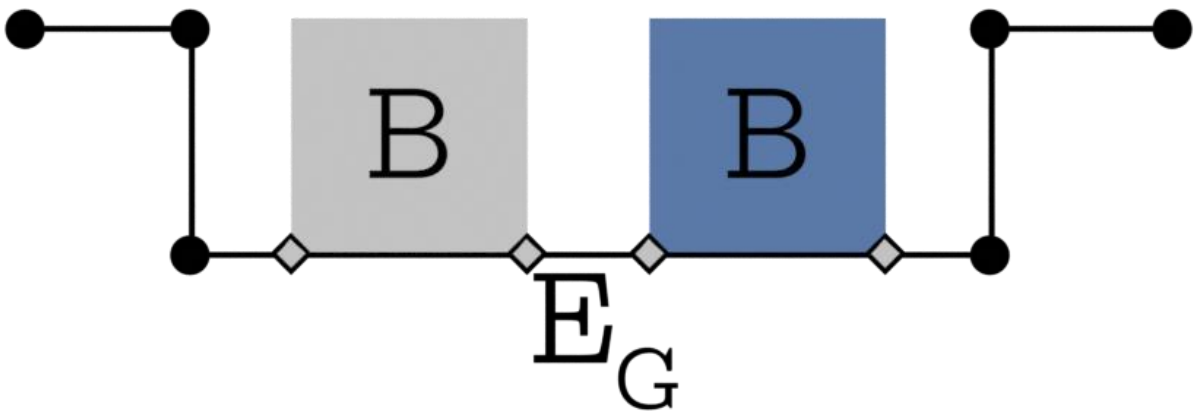
Implementation: Blocked Ledge Issue

- Undefined Case:
  How should we handle collisions on falledges?
- Simple solution: Just stop the fall → not nice for puzzles



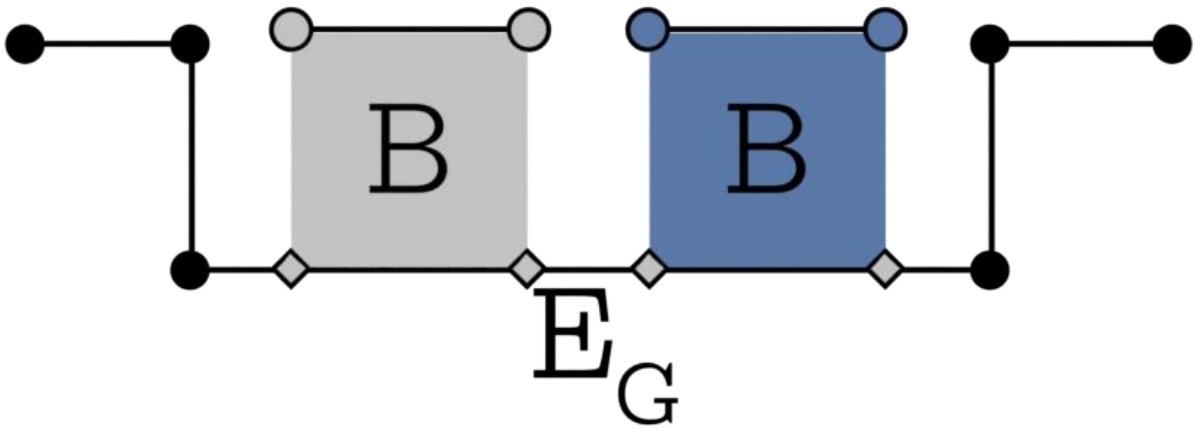
Implementation: Dynamic Graph

- Solution: Graphbodies stand ontop of each other



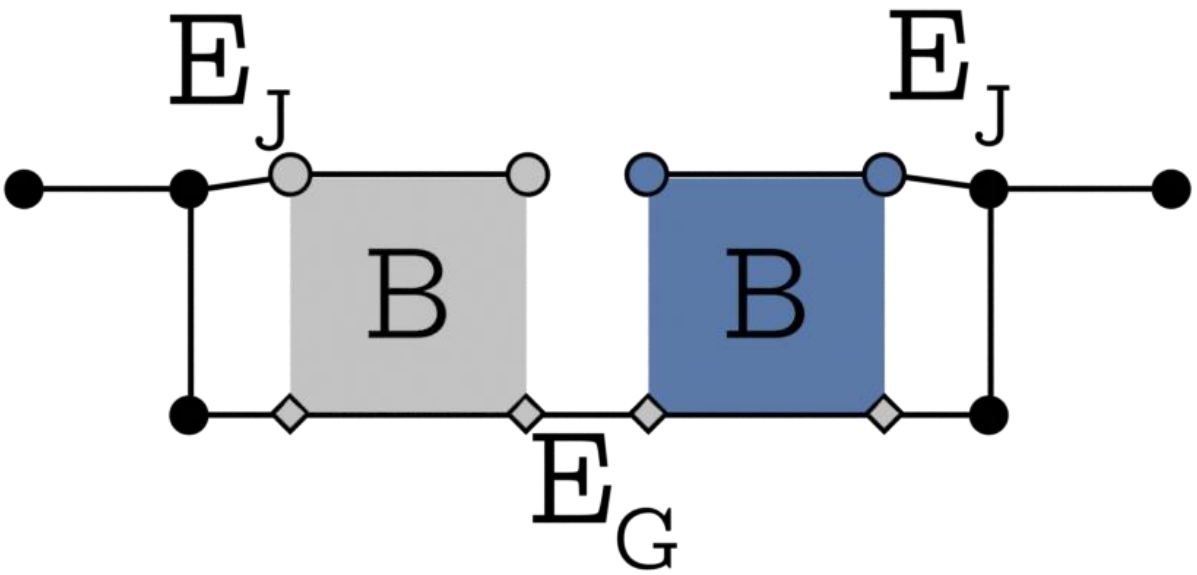
Implementation: Dynamic Graph

- Solution: Graphbodies stand ontop of each other
- Add vertices and edges to bodytops



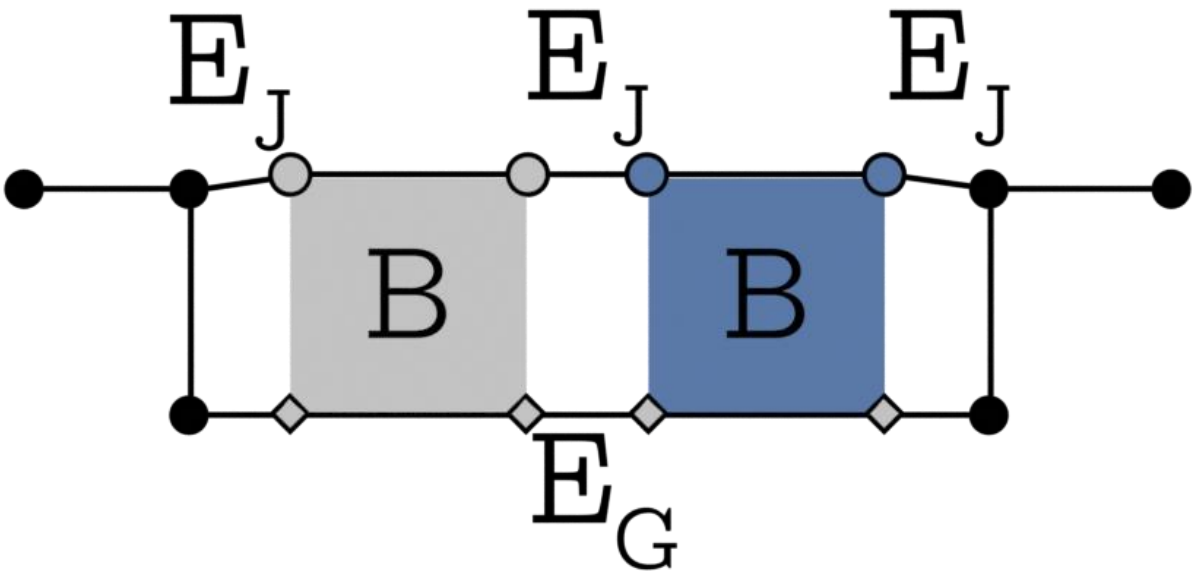
Implementation: Dynamic Graph

- Solution: Graphbodies stand ontop of each other
- Add vertices and edges to bodytops
- Connect bodytops and ledges with jumpedges



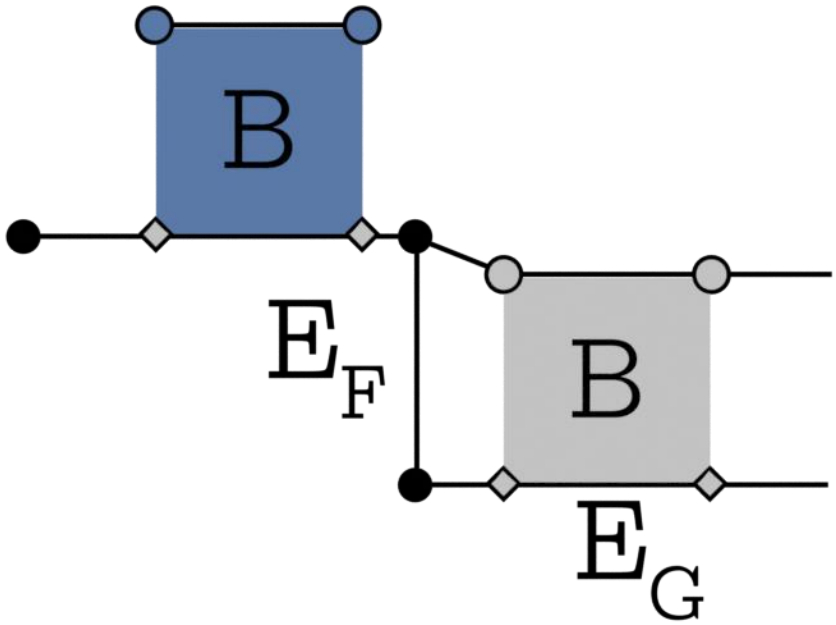
Implementation: Dynamic Graph

- Solution: Graphbodies stand ontop of each other
- Add vertices and edges to bodytops
- Connect bodytops and ledges with jumpedges
- Connect bodytops with bodytops with jumpedges



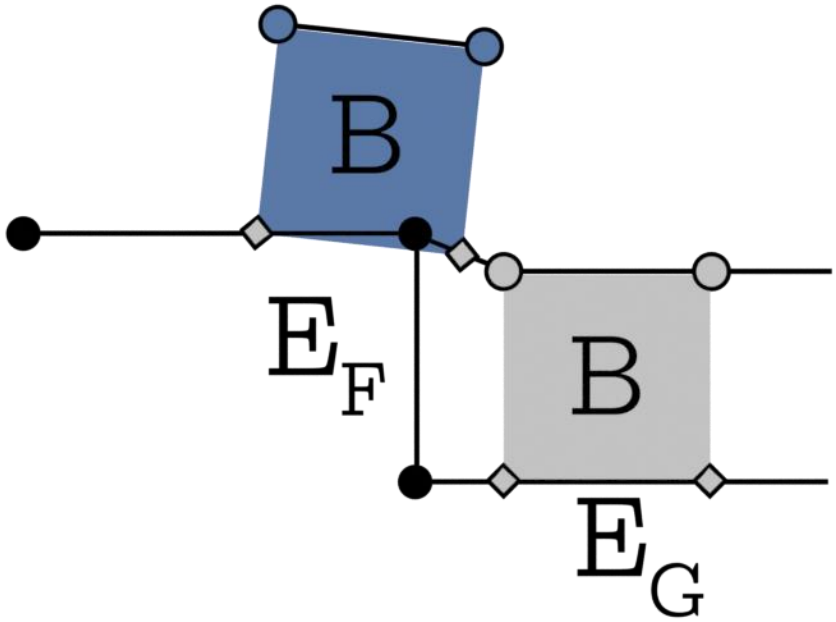
Implementation: Jumpedges

- Bodies should not stay on Jumpedges, since they change length



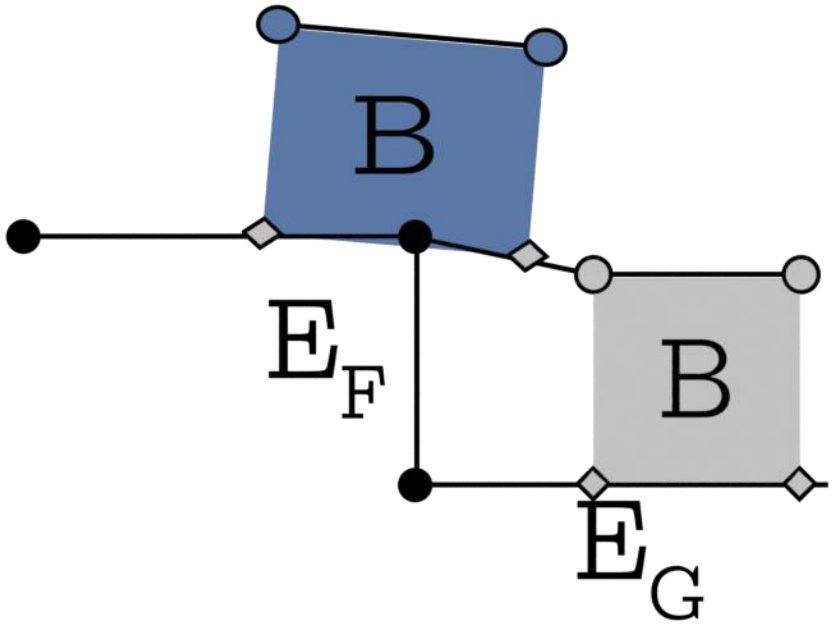
Implementation: Jumpedges

- Bodies should not stay on Jumpedges, since they change length



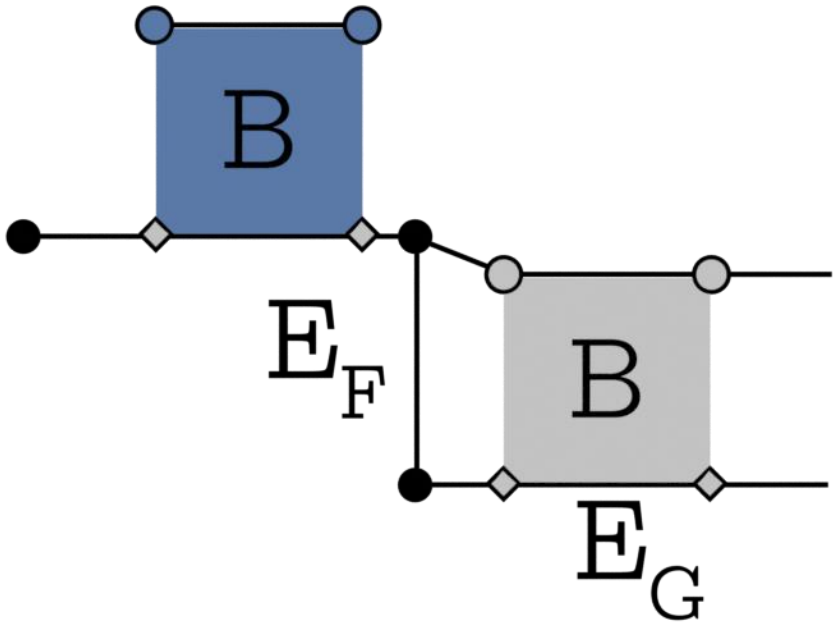
Implementation: Jumpedges

- Bodies should not stay on Jumpedges, since they change length



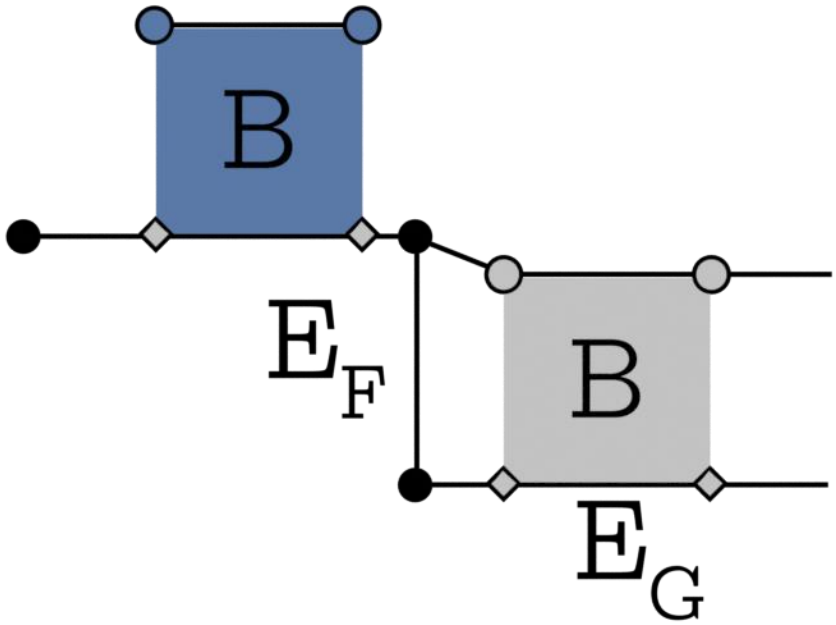
Implementation: Jumpedges

- Bodies should not stay on Jumpedges, since they change length



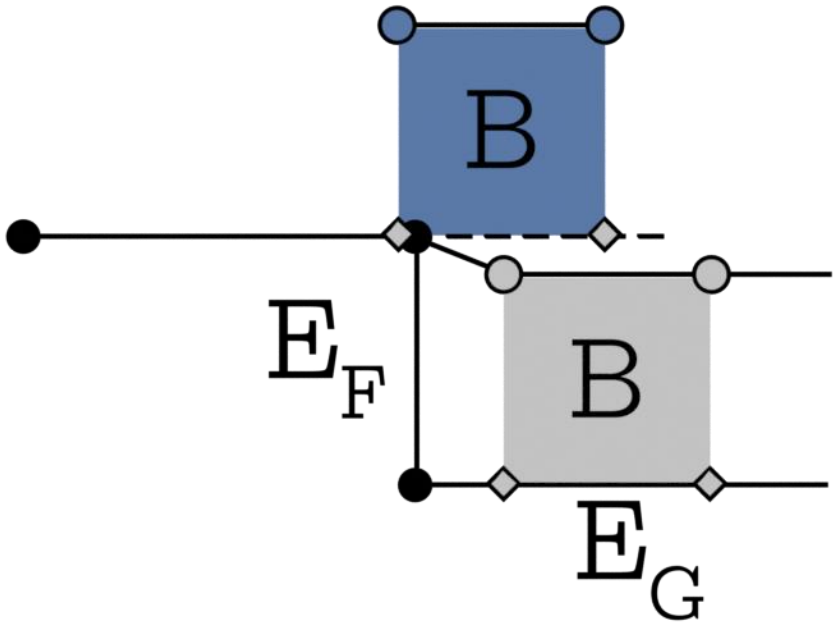
Implementation: Jumpedges

- Bodies should not stay on Jumpedges, since they change length



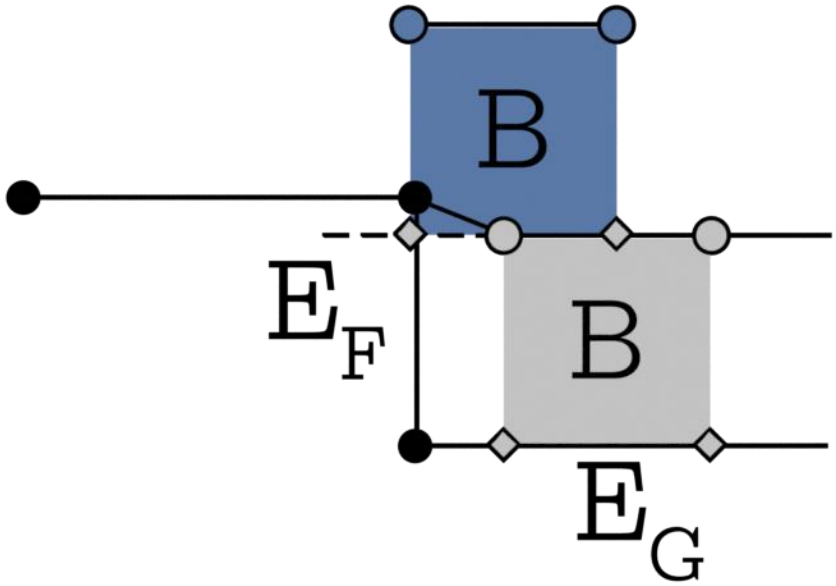
Implementation: Jumpedges

- Enter overhang as usual



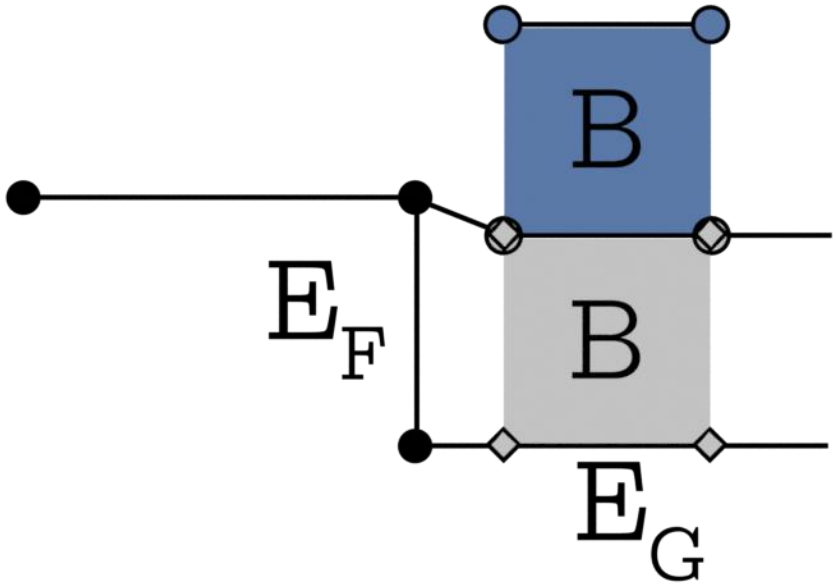
Implementation: Jumpedges

- Enter overhang as usual
- Jump is prioritized over fall (if jumpedge < width)
- Jump similar to fall, but use length of jumpedge as overhang

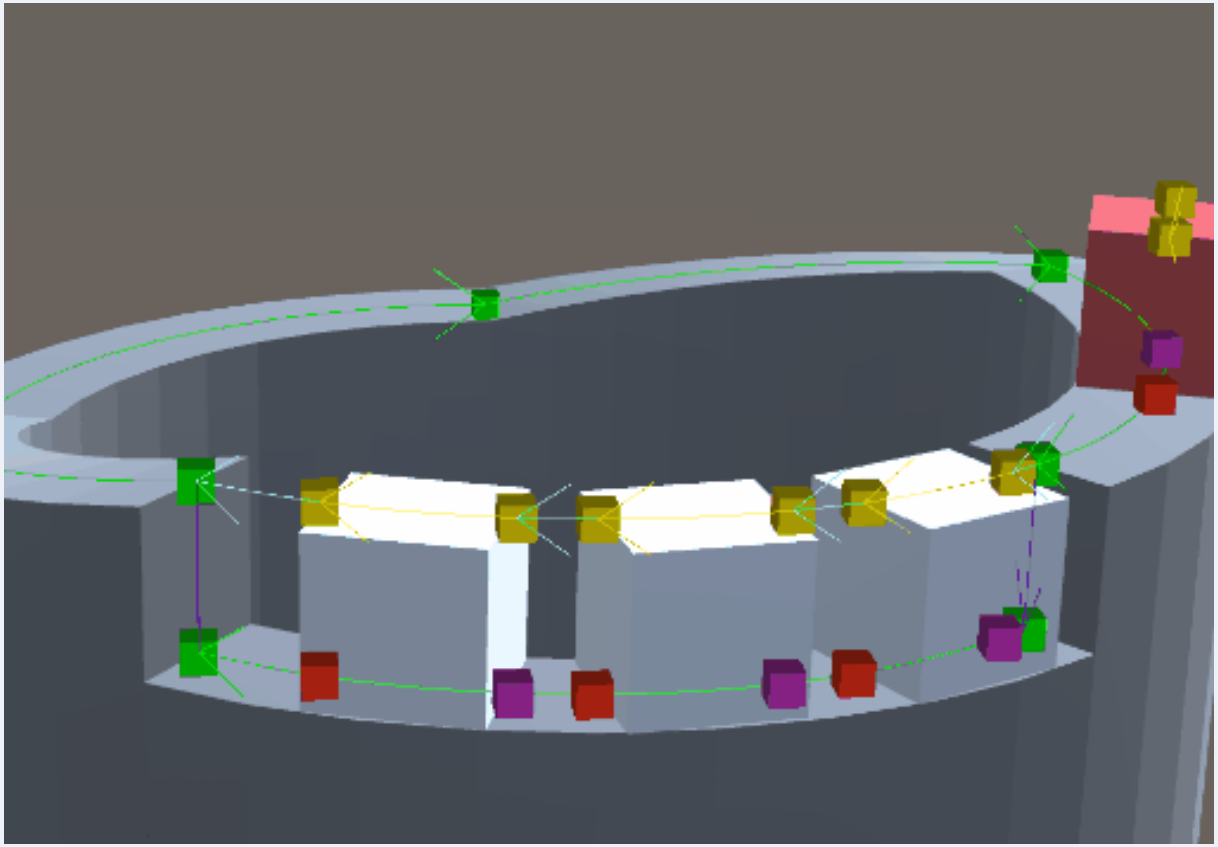


Implementation: Jumpedges

- Enter overhang as usual
- Jump is prioritized over fall (if jumpedge < width)
- Jump similar to fall, but use length of jumpedge as overhang



Implementation: Jumpedges



Prototype

Implementation: Dynamic Graph Issues

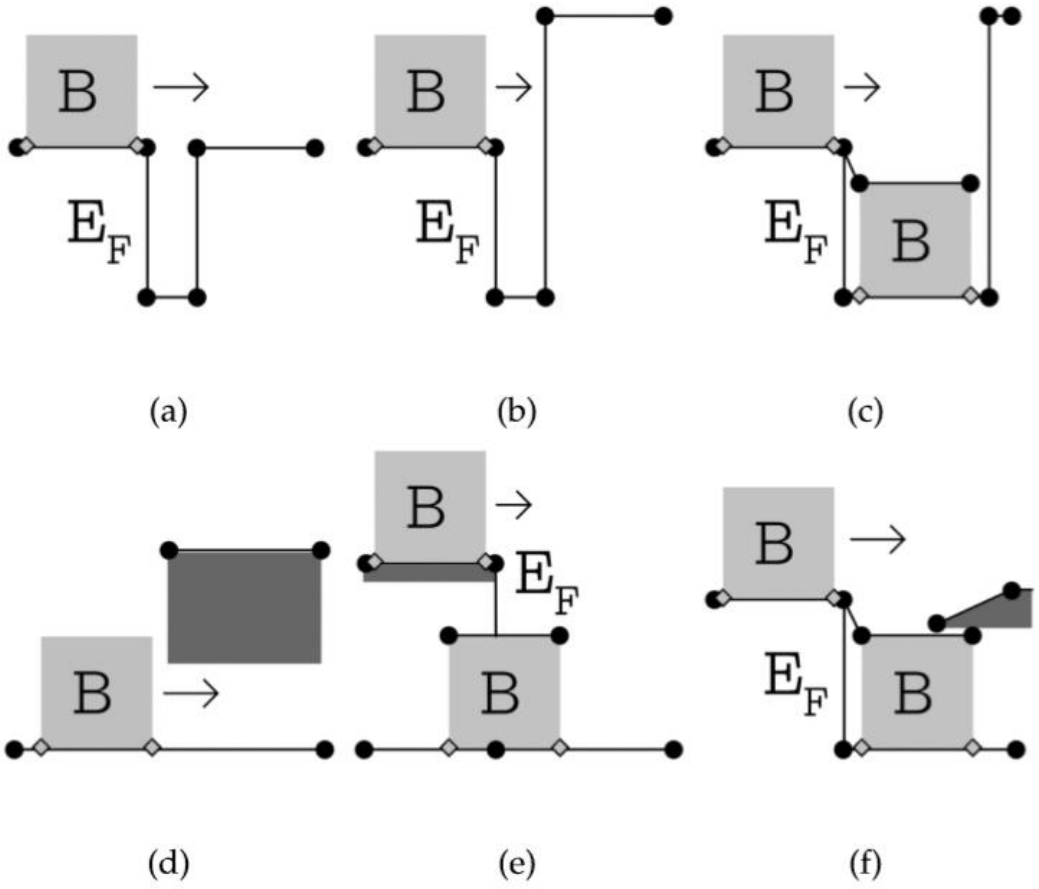
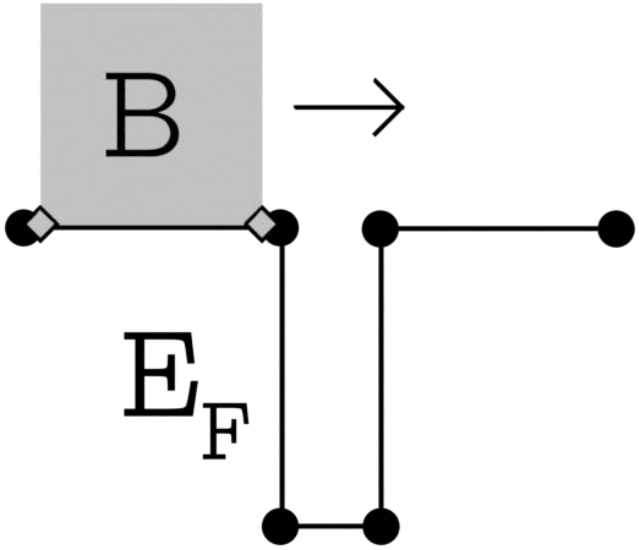


Figure 6.6: Unsupported Case Examples

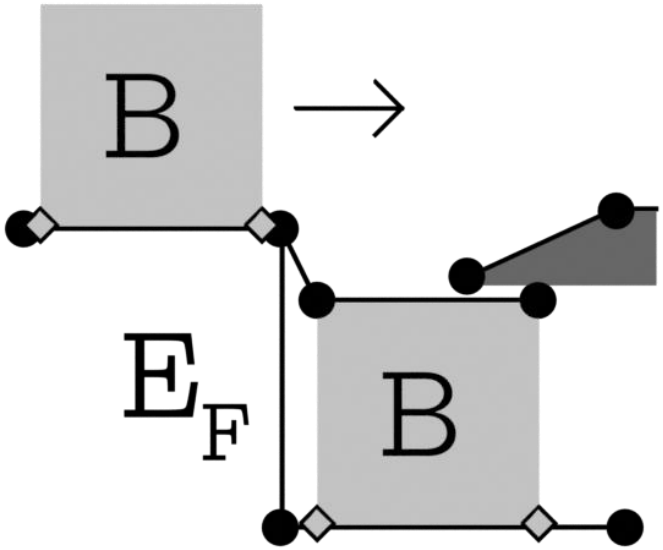
Implementation: Dynamic Graph Issues

- Simple Example:
  short pits
- Graphbody would fall on an edge that is to short
- Solution: simply add jumpedges to the top



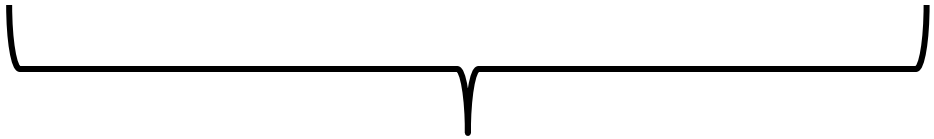
Implementation: Dynamic Graph Issues

- Complex Example:
  sharp ledges
- Graphbody can not jump from the middle of bodytop
- No simple solution



Implementation: Dynamic Graph Issues

falling, collisions, non 0 width

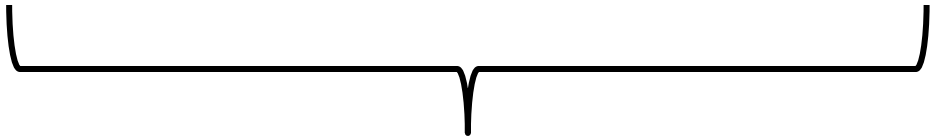


dynamic graph, jumpedges

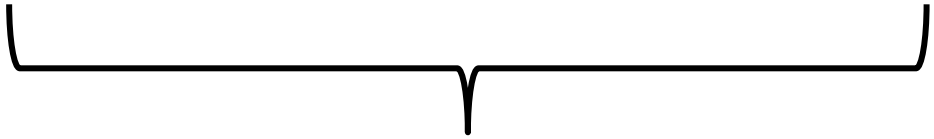
- No falledges → no ledges → no dynamic graph
- No collisions → no blocked ledges → no dyn graph
- 0 width → GB stay at the same place → no dyn graph

Implementation: Dynamic Graph Issues

falling, collisions, non 0 width

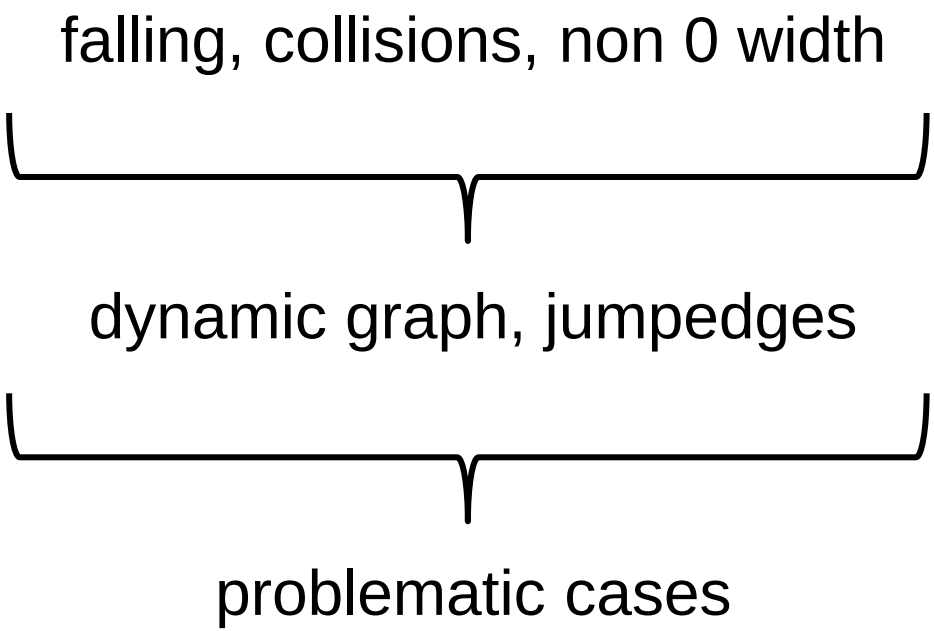


dynamic graph, jumpedges



problematic cases

Implementation: Dynamic Graph Issues



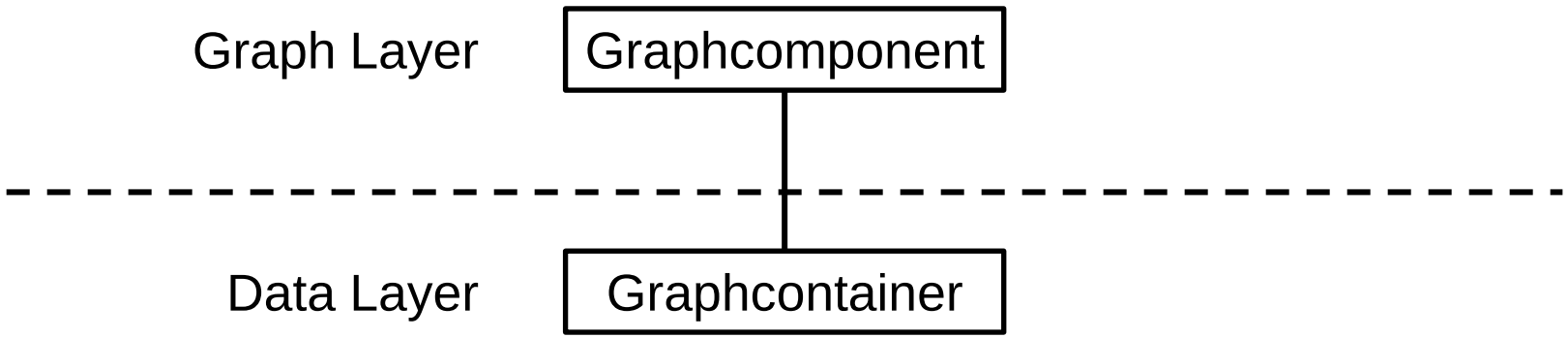
- If one of these three requirenments is not necessary, graphs are a good solution

Implementation: Architecture

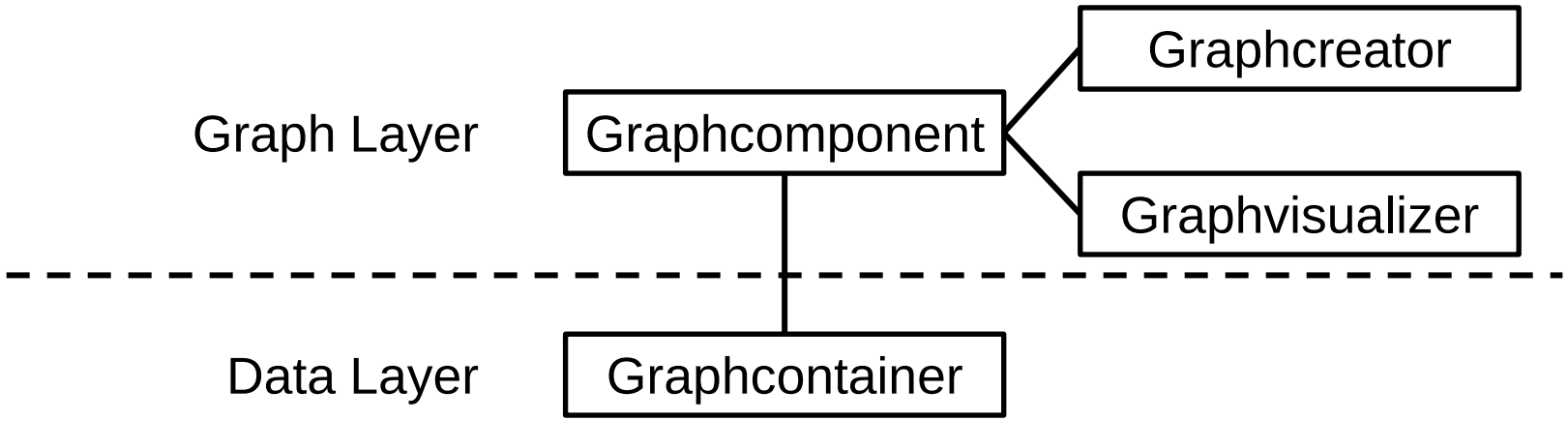
Data Layer

Graphcontainer

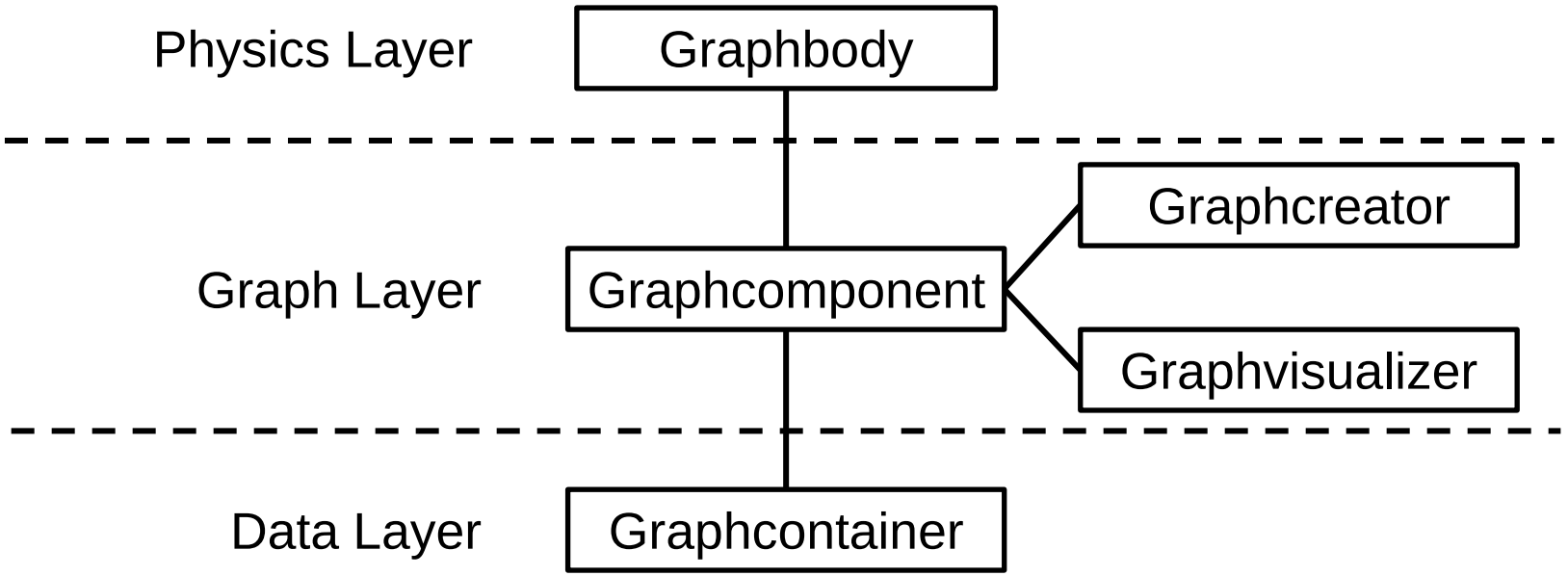
Implementation: Architecture



Implementation: Architecture

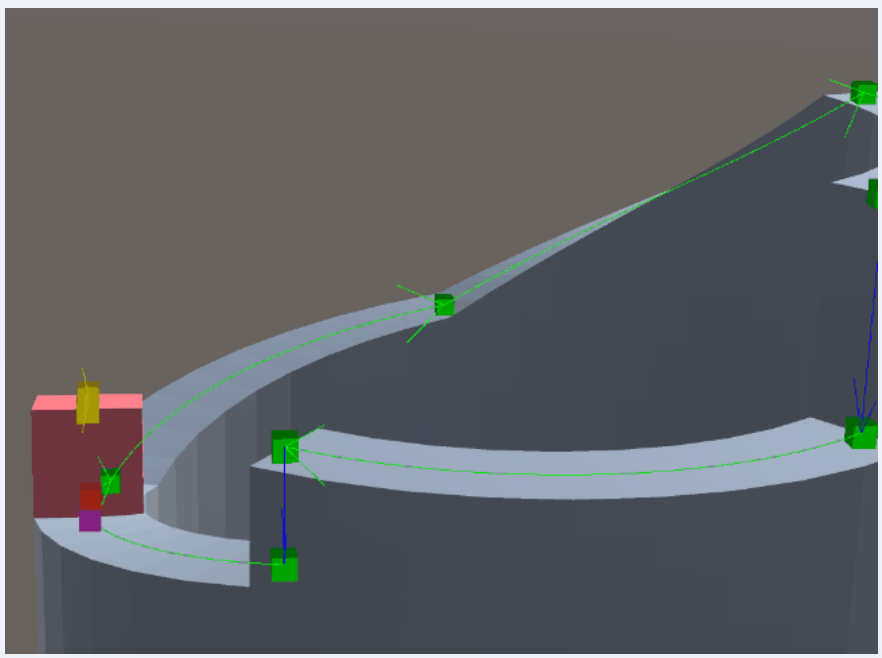


Implementation: Architecture



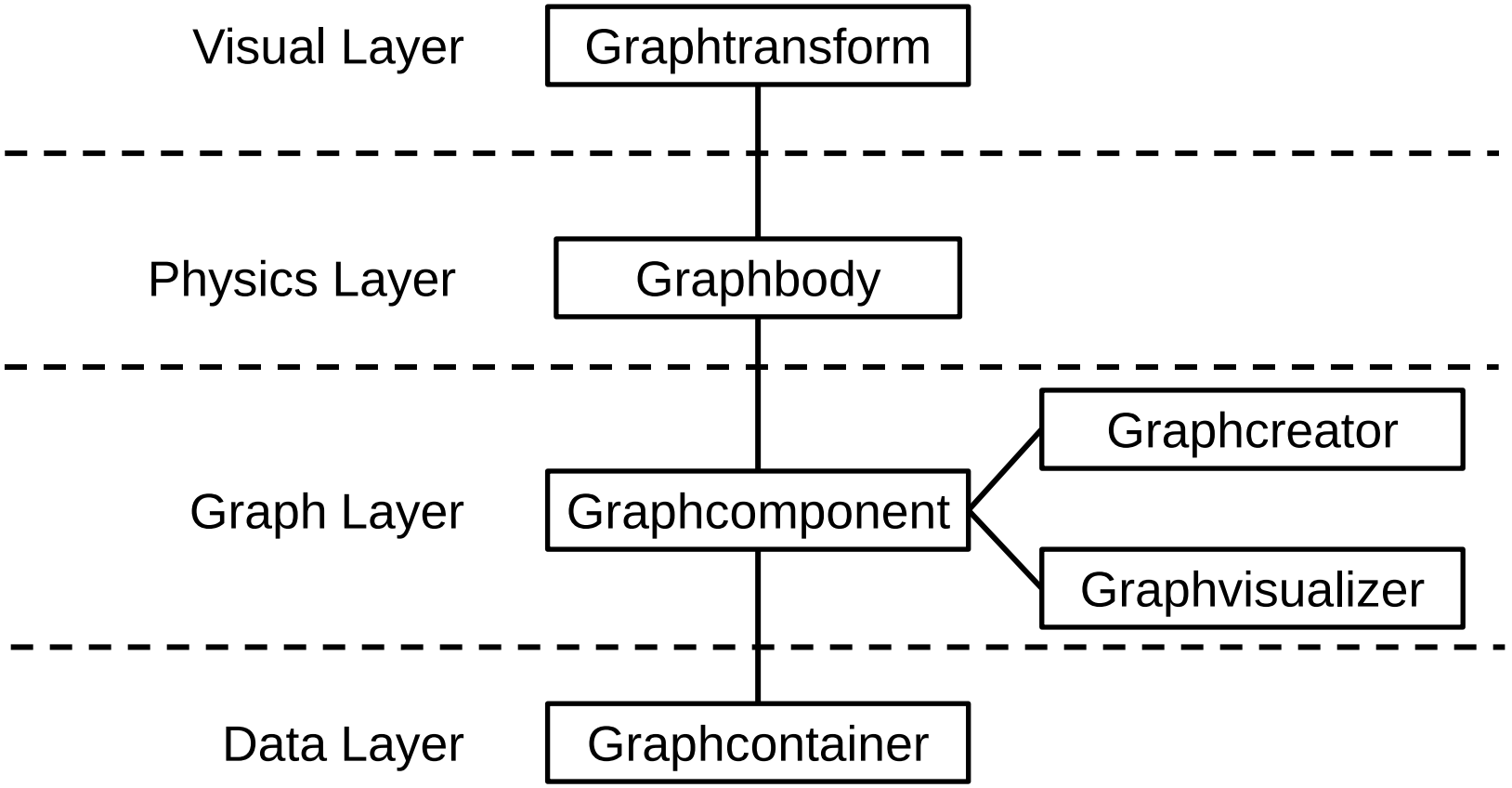
Exkurs: Physics

- Implemented friction and gravity
- Adjusted to make GB behave like Unitys Rigidbody2D

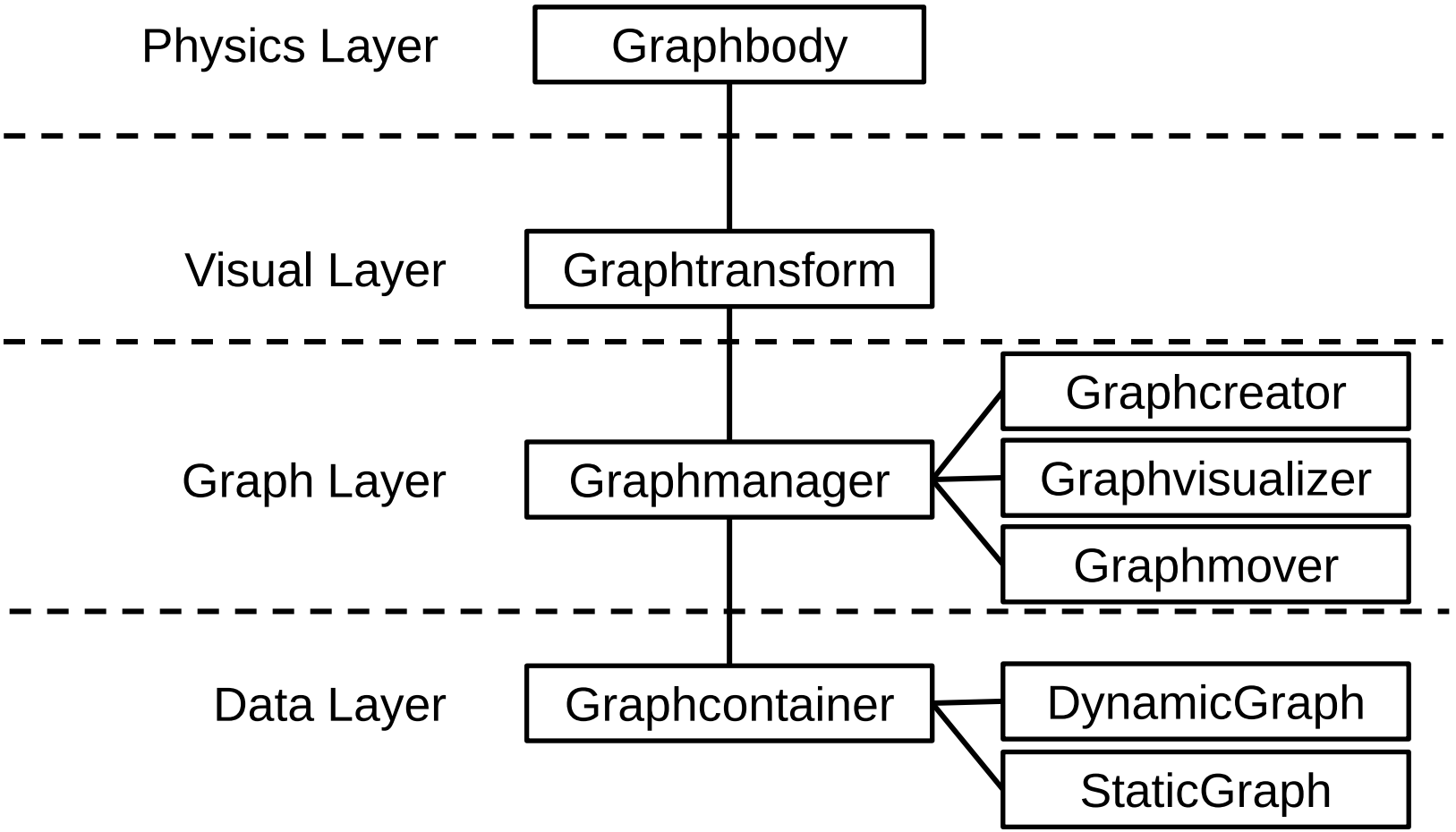


Prototype

Implementation: Architecture

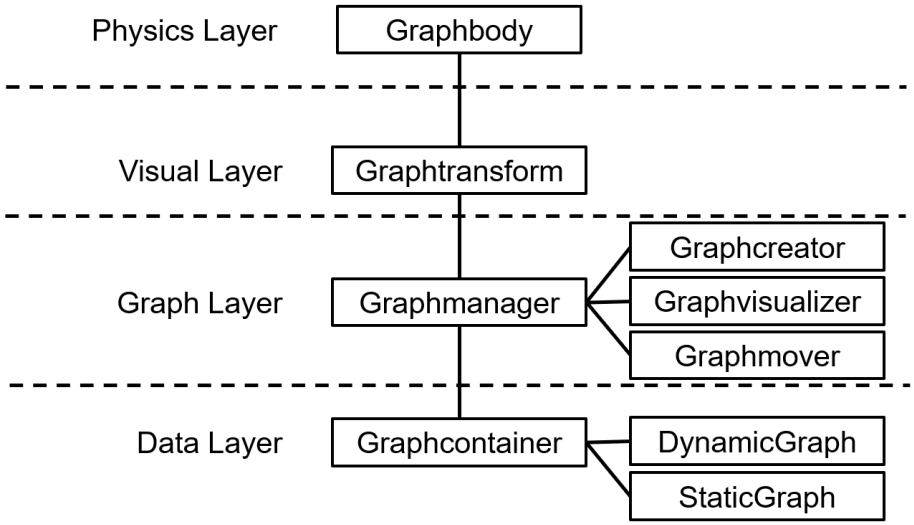


Implementation: optimized Architecture



Evaluation

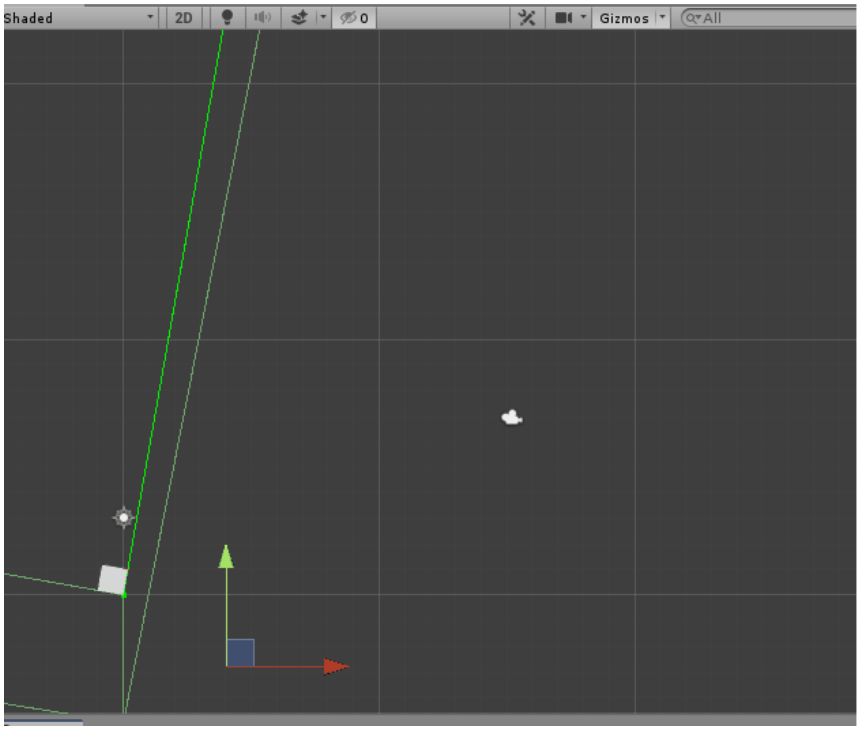
- Optimizing Architecture



- Decrease class size
- Modularity

Evaluation

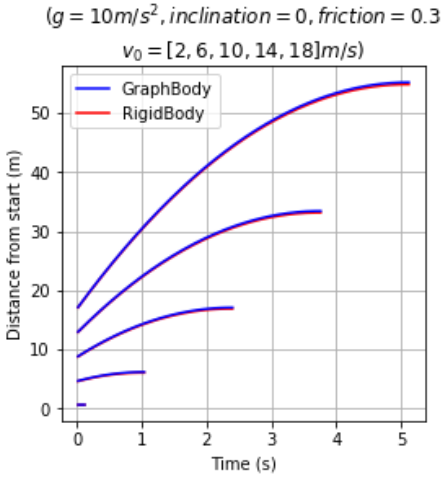
- Optimizing Architecture
- Physics Evaluation

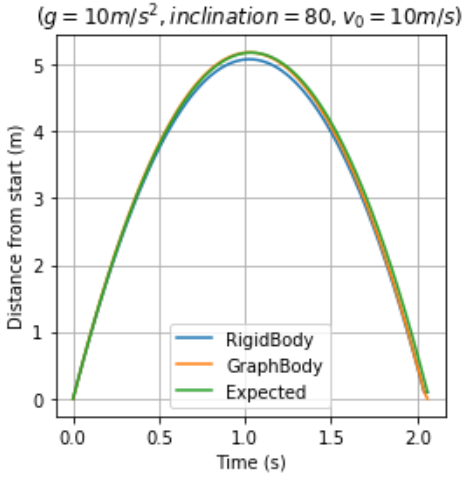


- Comparison between Rigidbody2D and Graphbody

Evaluation

- Optimizing Architecture
- Physics Evaluation

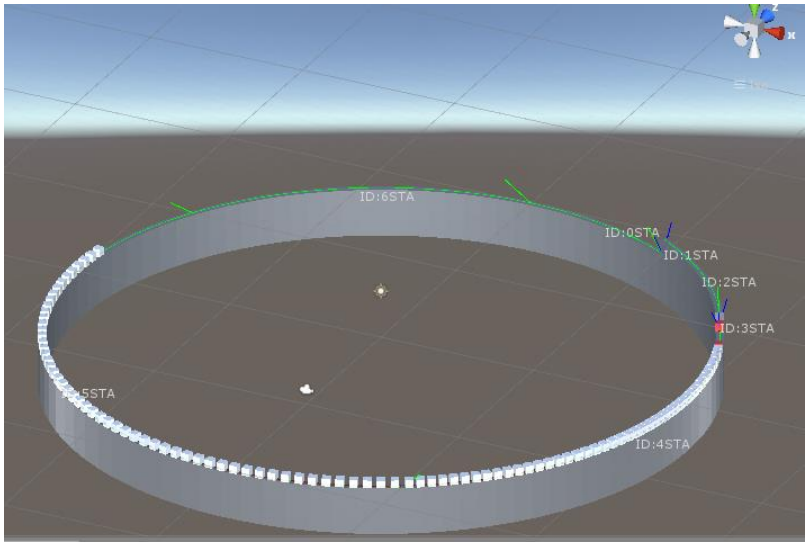




- Comparison between Rigidbody2D and Graphbody
- Separate Test for Friction and Gravity

Evaluation

- Optimizing Architecture
- Physics Evaluation
- Performance Evaluation



- Not well optimized
- Only measure growth
- Many boxes moving simultaniously is problematic

Evaluation

- Optimizing Architecture
- Physics Evaluation
- Performance Evaluation
- Usability & Predictability

- Still bad usability, but improved with more encapsulation
- Predictable and stable

Conclusion

- Graphbodies for Games
  - Current approach hardly usable in more complex scenarios
  - Static graph might be a good solution for other games
  - Consider using Rigidbody (stabilizing if necessary)

Conclusion

- Graphbodies for Games
  - Current approach hardly usable in more complex scenarios
  - Static graph might be a good solution for other games
  - Consider using Rigidbody (stabilizing if necessary)

- Further Research
  - More detailed overview over the possible designs
  - Other collision approaches

List of References

1. K. Wong. The Art of Monument Valley. url: https://youtu.be/i0X8-5PpYVg? t=1339
2. Broken Rules, Old Man’s Journey, 2018