Motion Planning using Hierarchical Graphs and Localized Tessellation of Two-Dimensional Space

Simon Pirsch

18.11.2021

Final: Bachelor Informatik - Games Engineering

Supervisor: Prof. Gudrun Klinker, Ph.D.

Advisor: Daniel Dyrda, M.Sc

Motivation – Video Games

- Days Gone: https://www.youtube.com/watch?v=bGej8K1r8KI

https://blog.playstation.com/2019/04/25/fighting-the-overwhelming-hordes-of-days-gone/

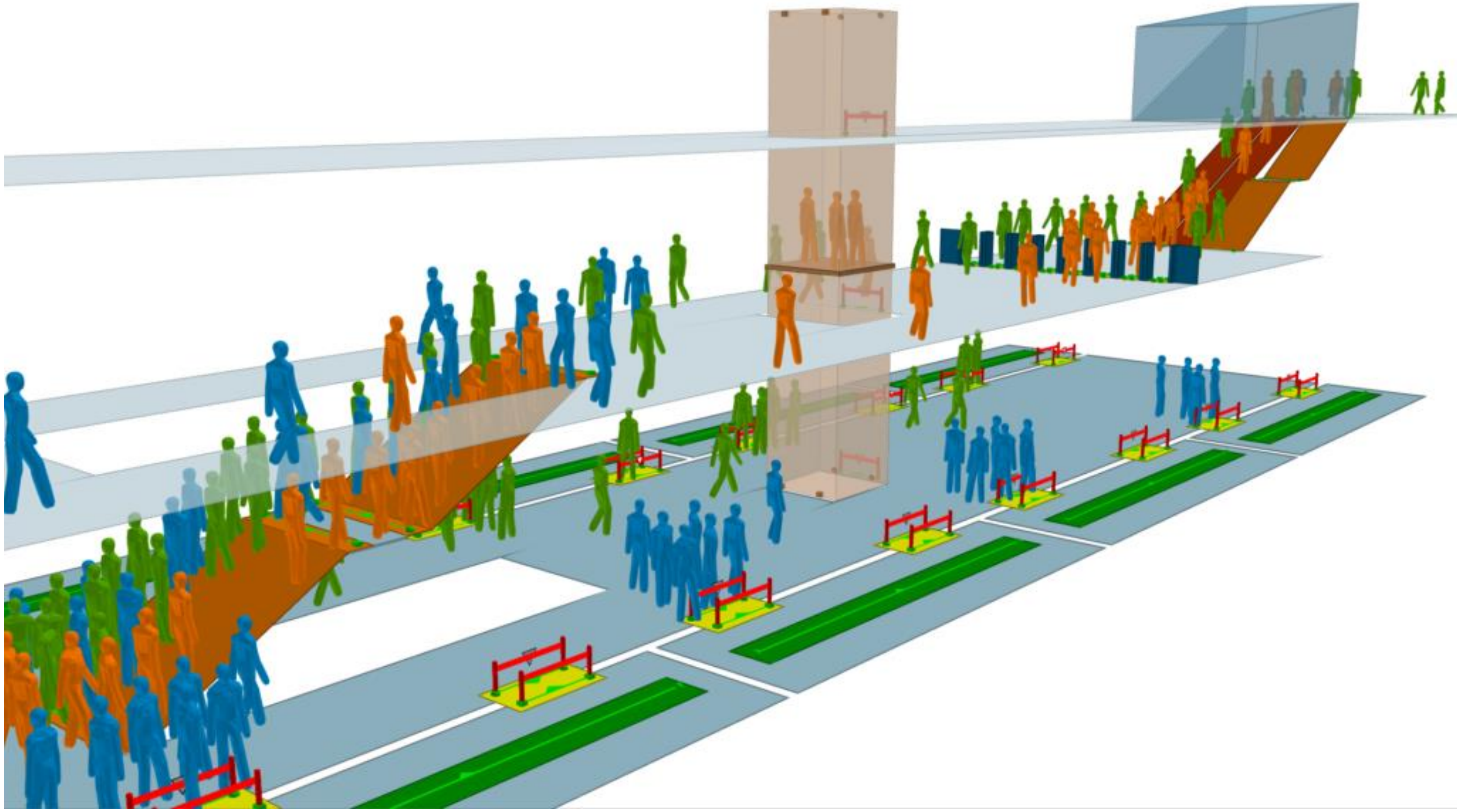
Motivation – Real-Time Strategy

- Age of Darkness: https://www.youtube.com/watch?v=rc3PzCoUd0Y

https://www.pcgamesn.com/age-of-darkness-final-stand/steam-early-access

Motivation – Crowd Simulation

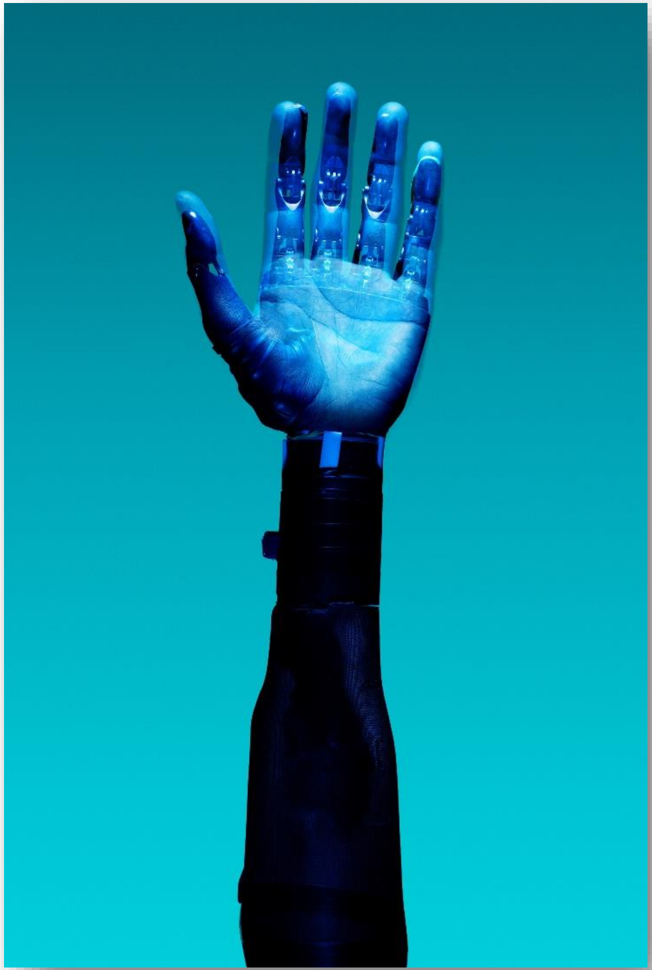
- MassMotion: https://www.oasys-software.com/products/pedestrian-simulation/massmotion/



https://www.oasys-software.com/news/launch-of-massmotion-multi-language/

Introduction

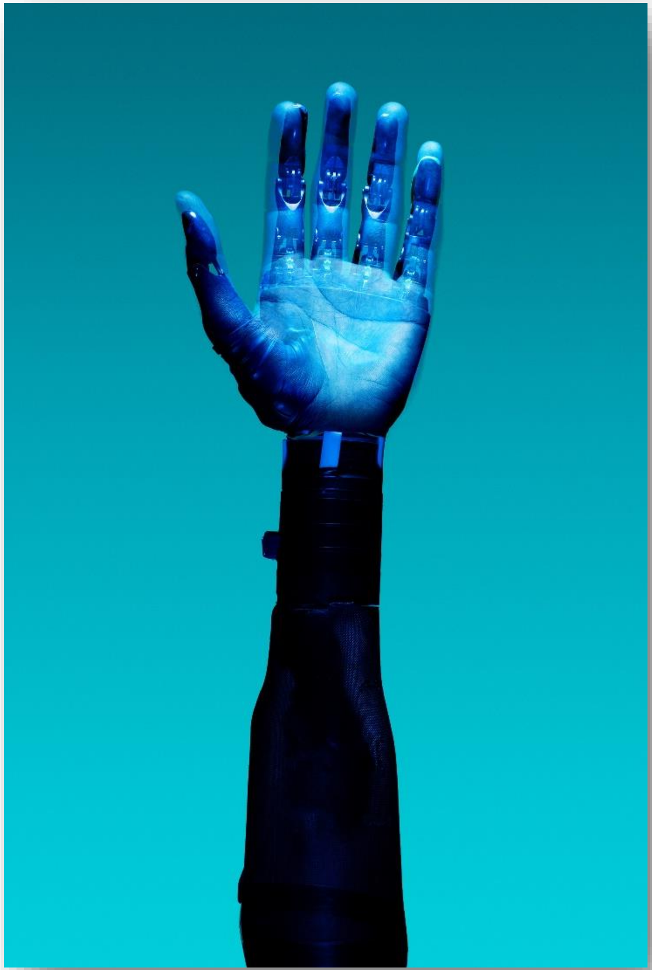
- **Motion Planning**
- **Hierarchical Graphs**
- **Localized Tessellation 2D**



https://www.pexels.com/photo/prosthetic-arm-on-blue-background-3913025/

Introduction

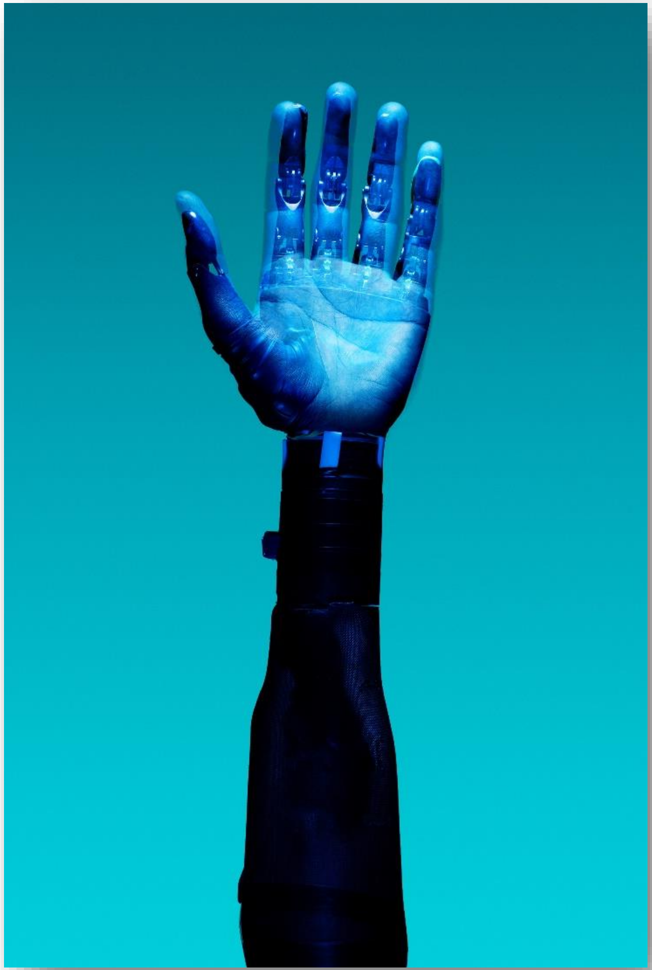
- **Motion Planning**
  - Preferred over “Pathfinding”
  - More than finding the shortest path
  - Include principles of Robotics
- **Hierarchical Graphs**
- **Localized Tessellation 2D**



https://www.pexels.com/photo/prosthetic-arm-on-blue-background-3913025/

Introduction

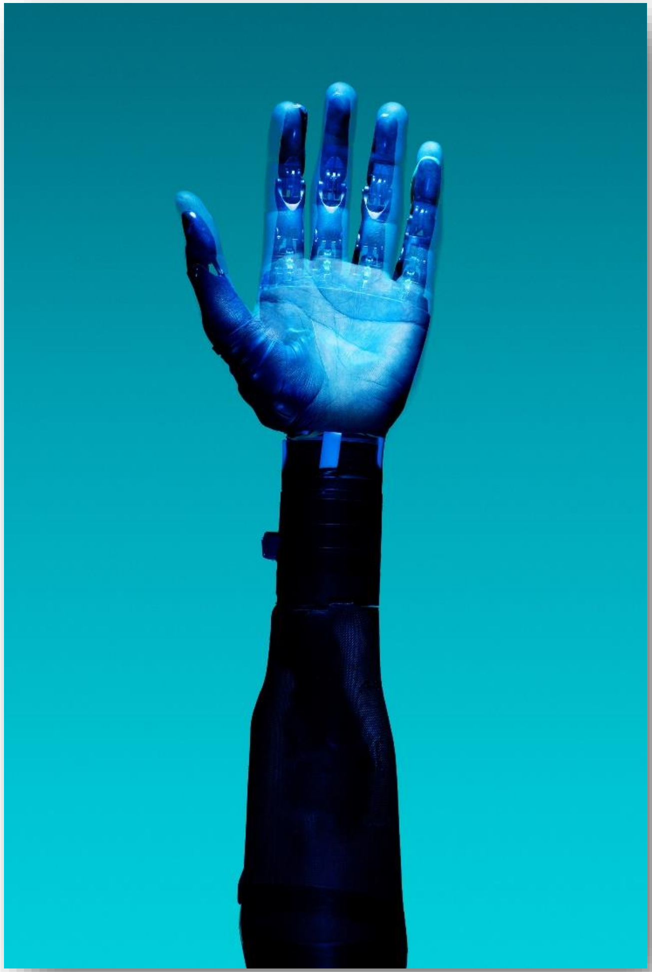
- **Motion Planning**
  - Preferred over “Pathfinding”
  - More than finding the shortest path
  - Include principles of Robotics
- **Hierarchical Graphs**
  - Abstractions via hierarchy
  - Depth = Level of detail
  - Nodes are subgraphs
- **Localized Tessellation 2D**



https://www.pexels.com/photo/prosthetic-arm-on-blue-background-3913025/

Introduction

- **Motion Planning**
  - Preferred over “Pathfinding”
  - More than finding the shortest path
  - Include principles of Robotics
- **Hierarchical Graphs**
  - Abstractions via hierarchy
  - Depth = Level of detail
  - Nodes are subgraphs
- **Localized Tessellation 2D**
  - 3D Surfaces
  - Topological aware tessellation
  - Reduce search graph complexity



https://www.pexels.com/photo/prosthetic-arm-on-blue-background-3913025/

Requirements & Problem Statement

- Optimality and Performance:
  → Introduce path sub-optimality to improve performance

Requirements & Problem Statement

- Optimality and Performance:
  → Introduce path sub-optimality to improve performance

- Environment and Learning:
  → Static environment
  → No autonomous agent learning

Requirements & Problem Statement

- Optimality and Performance:
  → Introduce path sub-optimality to improve performance

- Environment and Learning:
  → Static environment
  → No autonomous agent learning

- Single- and Multi-Agent Environments:
  → Single-Agent path calculation with multi-agent context

Requirements & Problem Statement

- Optimality and Performance:
  → Introduce path sub-optimality to improve performance
- Environment and Learning:
  → Static environment
  → No autonomous agent learning
- Single- and Multi-Agent Environments:
  → Single-Agent path calculation with multi-agent context

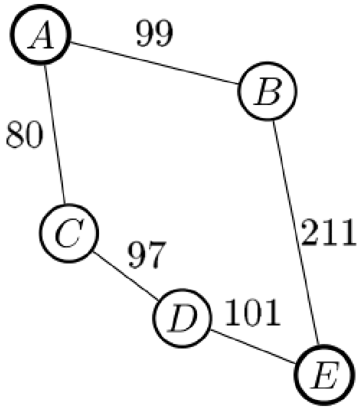
→ Scalability & Performance

Related Work

- Artificial Intelligence: “Solving Problems by Searching”

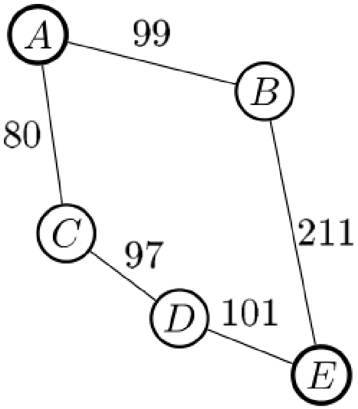
Related Work

- Artificial Intelligence: “Solving Problems by Searching”
- Search algorithms on graphs:
  - **Breadth-First Search**:
    - Frontier: FIFO Queue
    - Explored Set



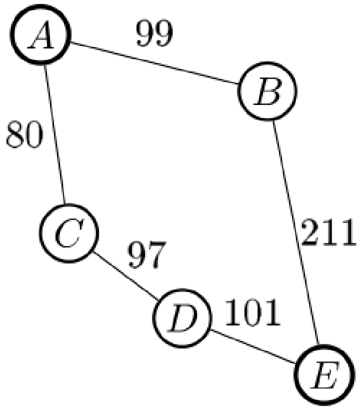
Related Work

- Artificial Intelligence: “Solving Problems by Searching”
- Search algorithms on graphs:
  - **Breadth-First Search**:
    - Frontier: FIFO Queue
    - Explored Set
  - **Uniform-Cost Search** (Dijkstra):
    - Frontier: Priority Queue
    - Ordering g: "$total\ path\ cost\ from\ start\ node$"



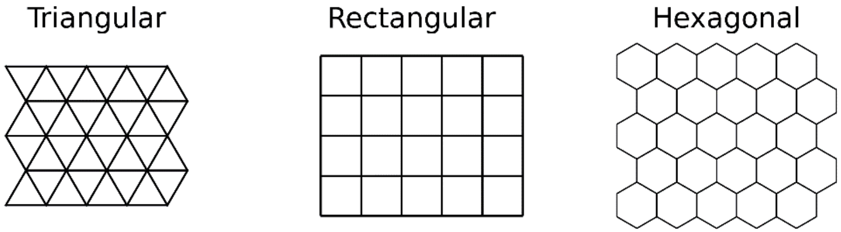
Related Work

- Artificial Intelligence: “Solving Problems by Searching”
- Search algorithms on graphs:
  - **Breadth-First Search**:
    - Frontier: FIFO Queue
    - Explored Set
  - **Uniform-Cost Search** (Dijkstra):
    - Frontier: Priority Queue
    - Ordering g: "$total\ path\ cost\ from\ start\ node$"
  - **A***:
    - Heuristic h: "$euclidean\ distance\ to\ goal\ node$"
    - Ordering $f = g + h$
    - Complexity: $\mathcal{O}\big((|E| + |V|)\log|V|\big)$



Regular Tessellation: Grid

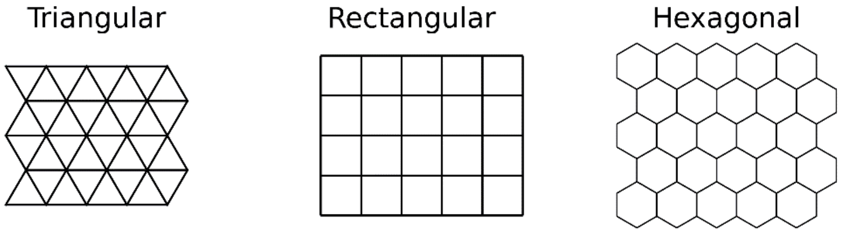
- Simple regular polygon:
  - No holes or self-intersection
  - Equilateral + Equiangular



\+ Simple storage, Uniformity

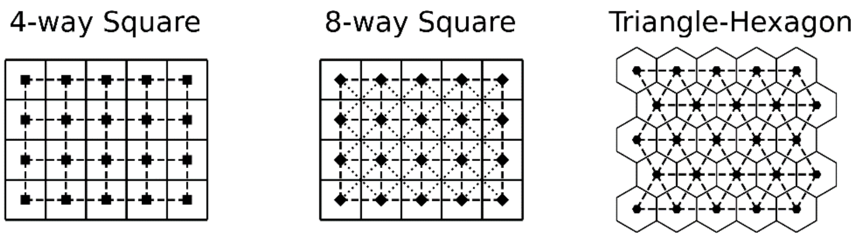
Regular Tessellation: Grid

- Simple regular polygon:
  - No holes or self-intersection
  - Equilateral + Equiangular

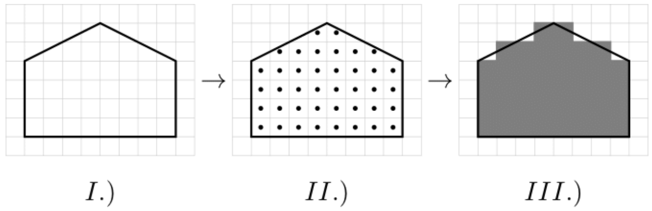


+ Simple storage, Uniformity

- Connectivity:
  - Shared edges: N_e
  - Shared vertices: N_v
  - Graph duality

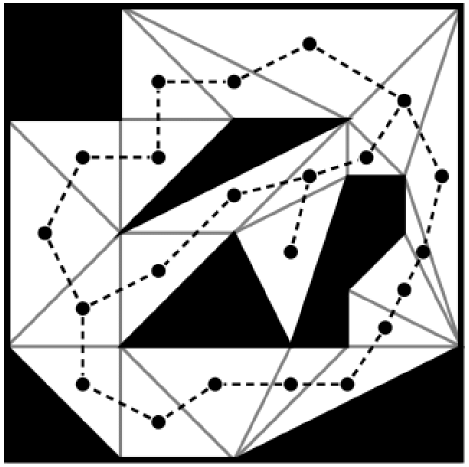


– Rasterization, Accuracy



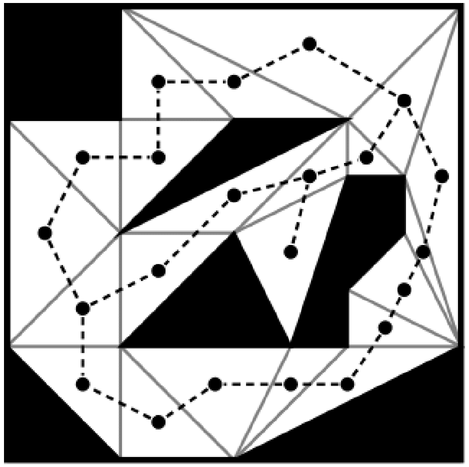
Irregular Tessellation: NavMesh

- Constrained Delaunay Triangulation (CDT):
  - Obstacle dense area → more triangles
  - Represent arbitrary complex polygons
  - Maximize minimal internal angle
  - Produces good search graph
  - Shared unconstrained edges: N_{ue}

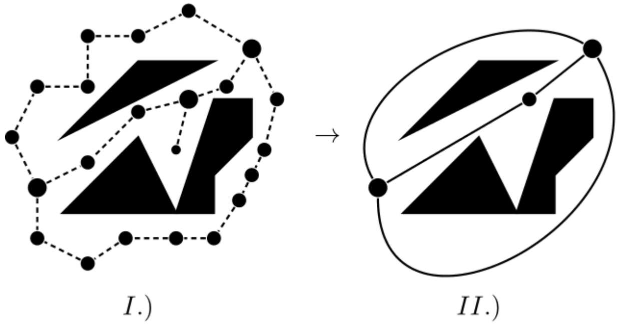


Irregular Tessellation: NavMesh

- Constrained Delaunay Triangulation (CDT):
  - Obstacle dense area → more triangles
  - Represent arbitrary complex polygons
  - Maximize minimal internal angle
  - Produces good search graph
  - Shared unconstrained edges: N_{ue}



- Search (multi-)graph reduction:

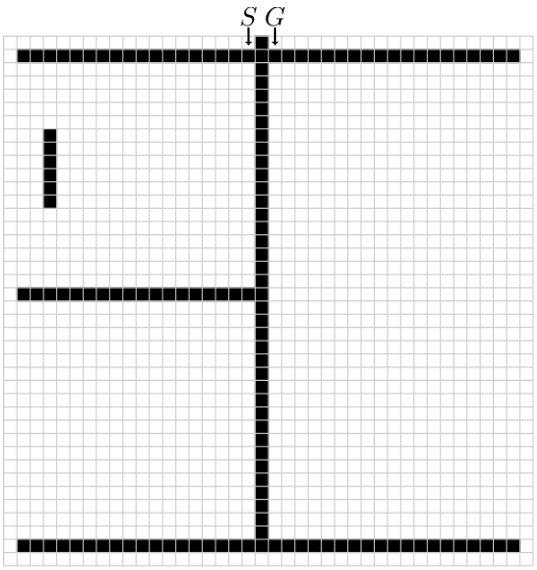


Hierarchical Graphs – HPA*

- Hierarchical Path-Finding A*

- Square grid based

- Example:
  - 40×40 maze

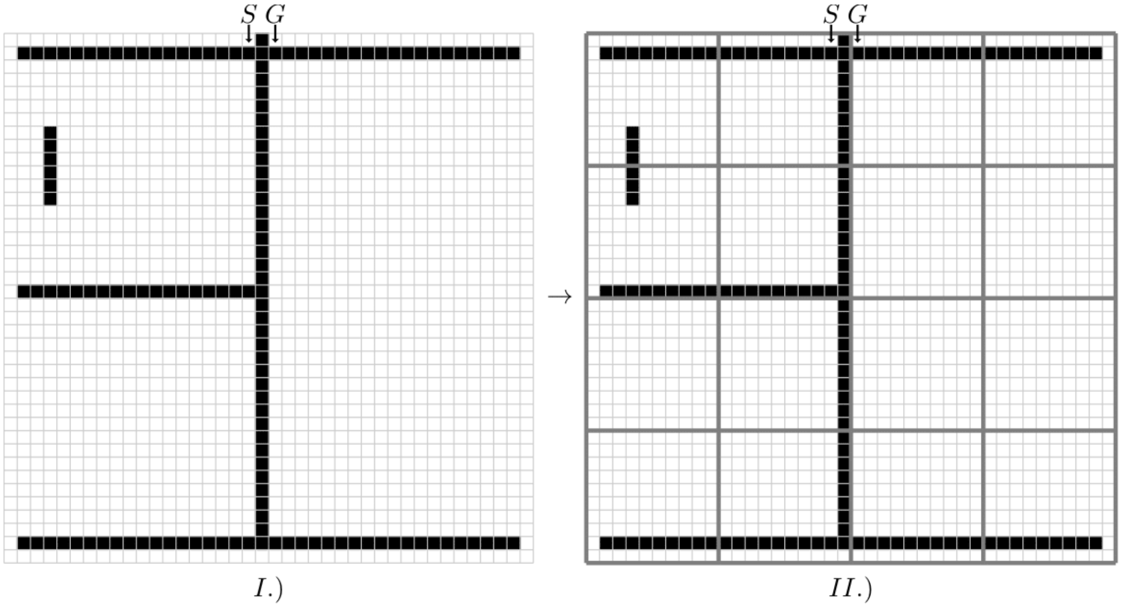


$I.)$

- Initial maze: $I.)$

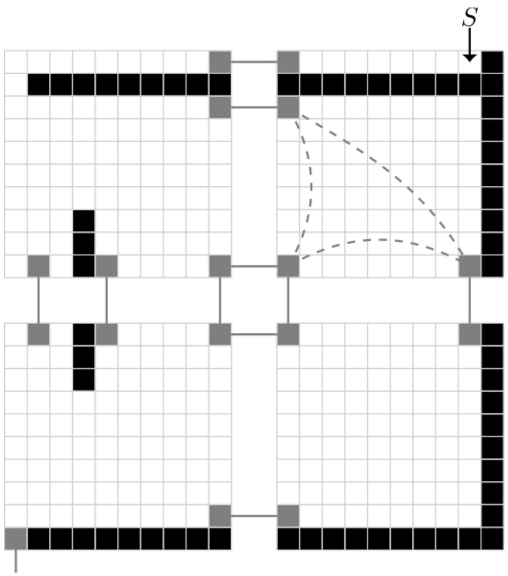
Hierarchical Graphs – HPA*

- Hierarchical Path-Finding A*
- Square grid based
- Example:
  - 40×40 maze
  - 16 10×10 clusters
- Initial maze: $I.)$
- Global level of clusters: $II.)$



HPA*: Linked Local Clusters

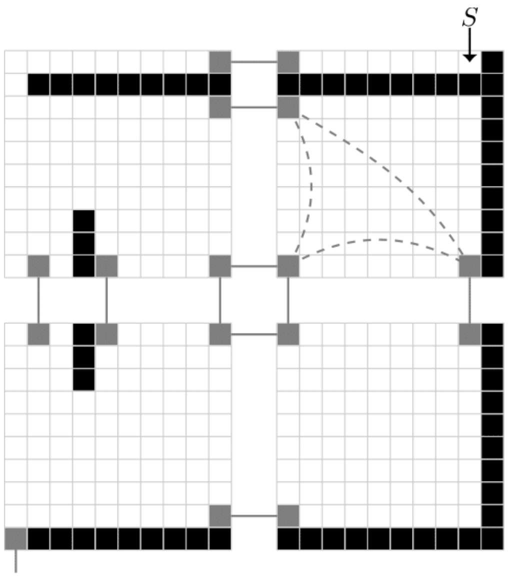
Inter-edges:



Constant cost: 1

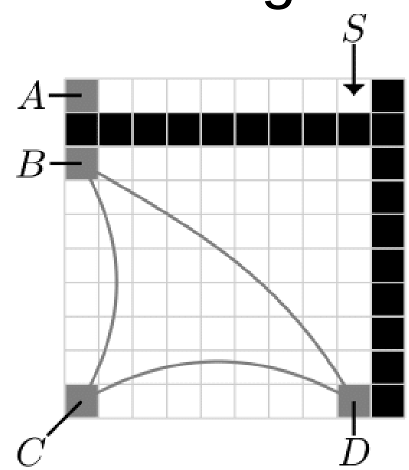
HPA*: Linked Local Clusters

Inter-edges:



Constant cost: 1

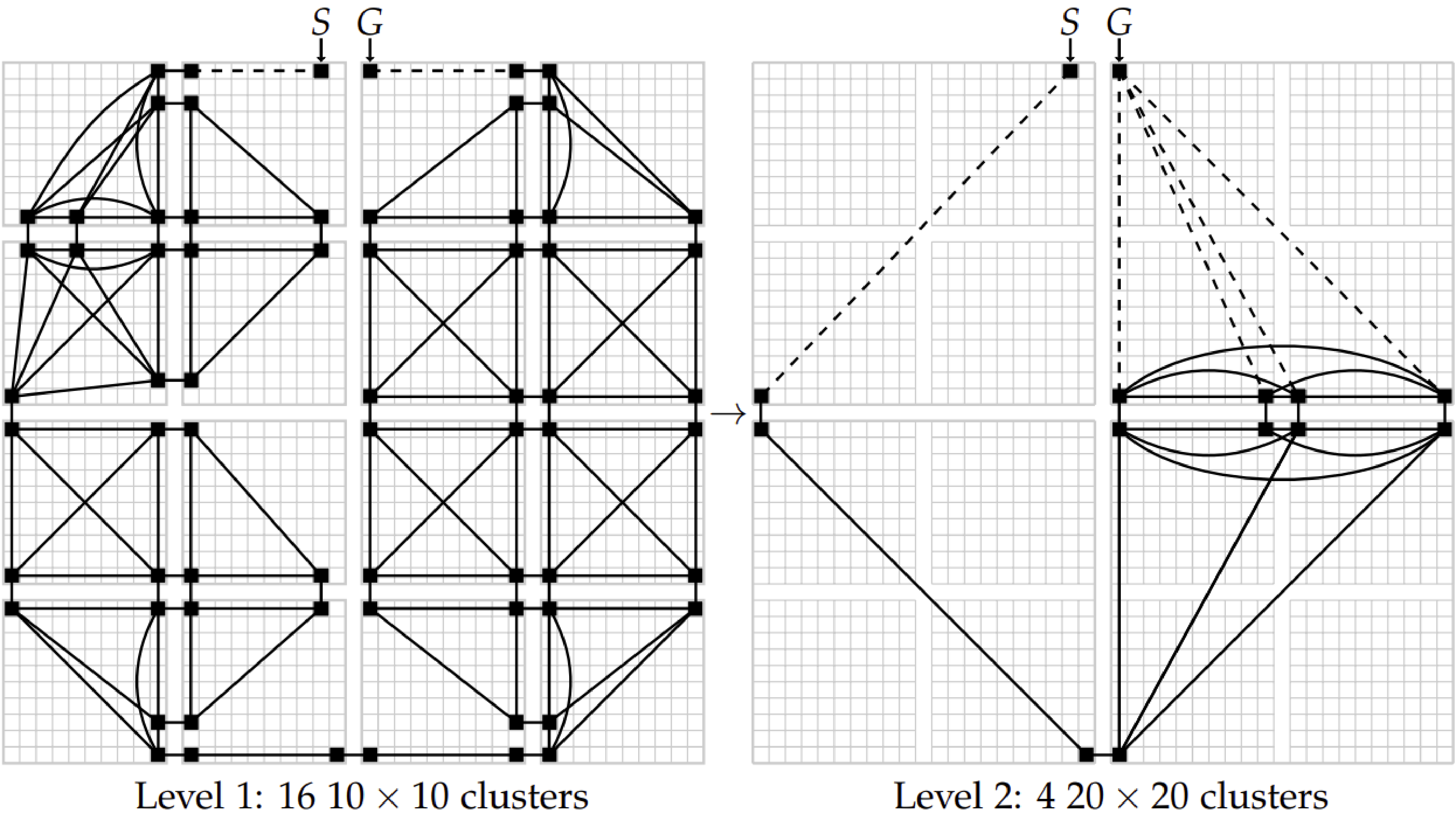
Intra-edges:



Calculated costs:

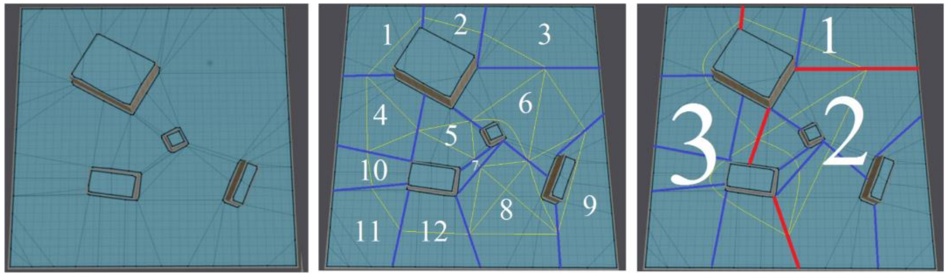
	A	*B*	*C*	*D*
A		∞	∞	∞
B			7.00	10.94
C				8.00
D				

HPA*: Abstraction Levels



Hierarchical Graphs – HNA*

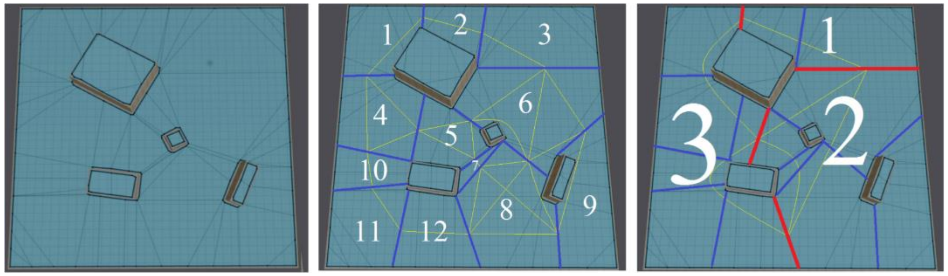
- Hierarchical NavMesh Path-finding algorithm



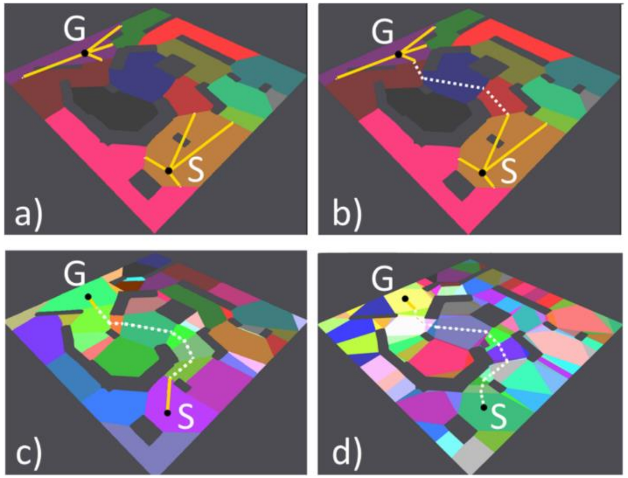
- NavMesh based
  - Convex polygons
  - Initial graph $G_0 = (V_0, E_0)$
  - Recursively partition η nodes $G_i = (V_i, E_i)$
  - Upper limit $0 < i \leq m \in \{\lceil \log_\eta |V_0| \rceil, L_{max}\}$

Hierarchical Graphs – HNA*

- Hierarchical NavMesh Path-finding algorithm



- NavMesh based
  - Convex polygons
  - Initial graph $G_0 = (V_0, E_0)$
  - Recursively partition η nodes $G_i = (V_i, E_i)$
  - Upper limit $0 < i \leq m \in \{\lceil \log_\eta |V_0| \rceil, L_{max}\}$

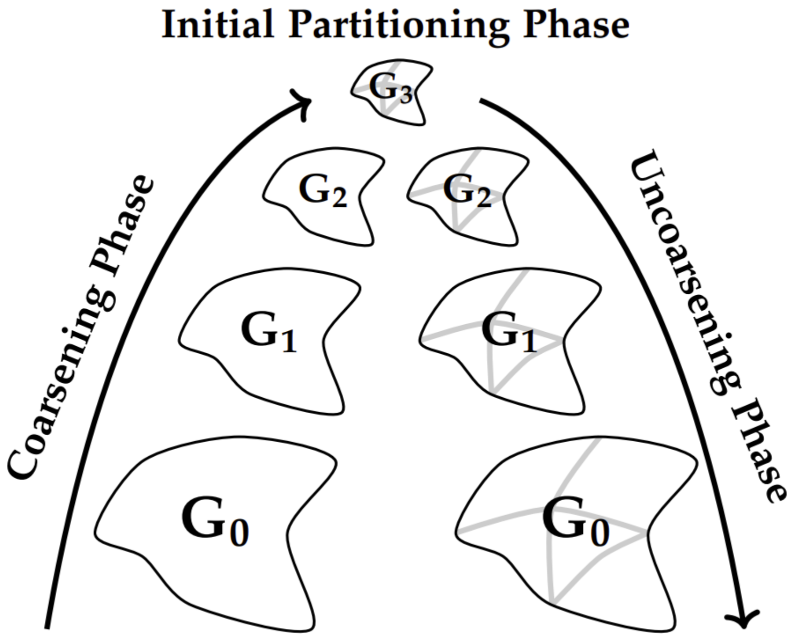


- MultiLevel k-way Partitioning (MLkP)
  - Balanced amount of components
  - Reduce edges between components
  - METIS

HNA*: MLkP Phases

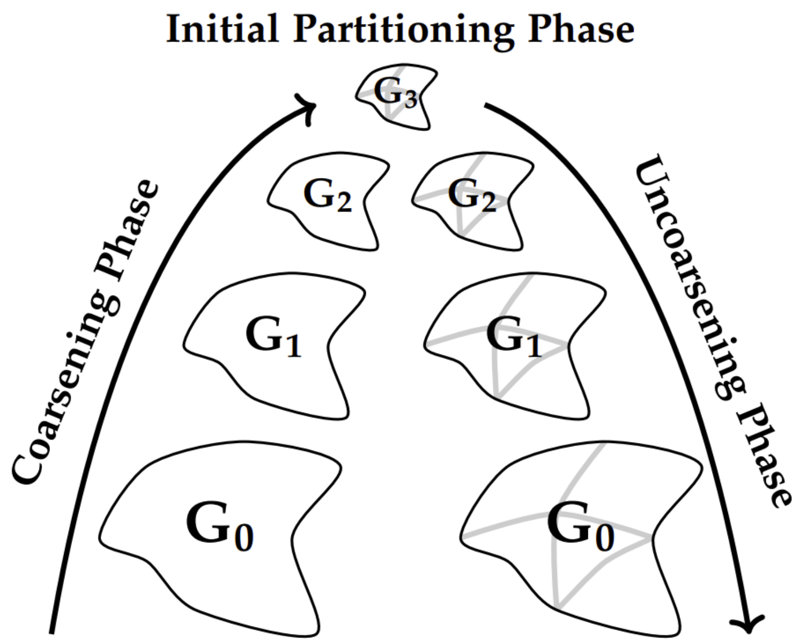
- Coarsening Phase:

  recursively collapse nodes into smaller graphs G_{i+1}

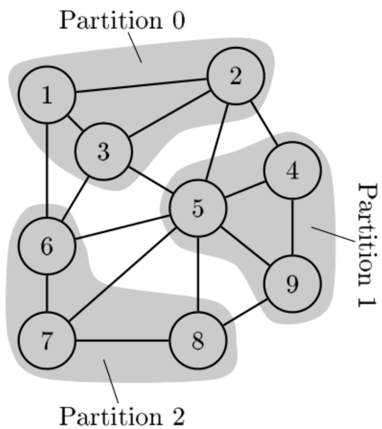


HNA*: MLkP Phases

- Coarsening Phase:
  recursively collapse nodes into smaller graphs G_{i+1}



- Initial Partitioning Phase:

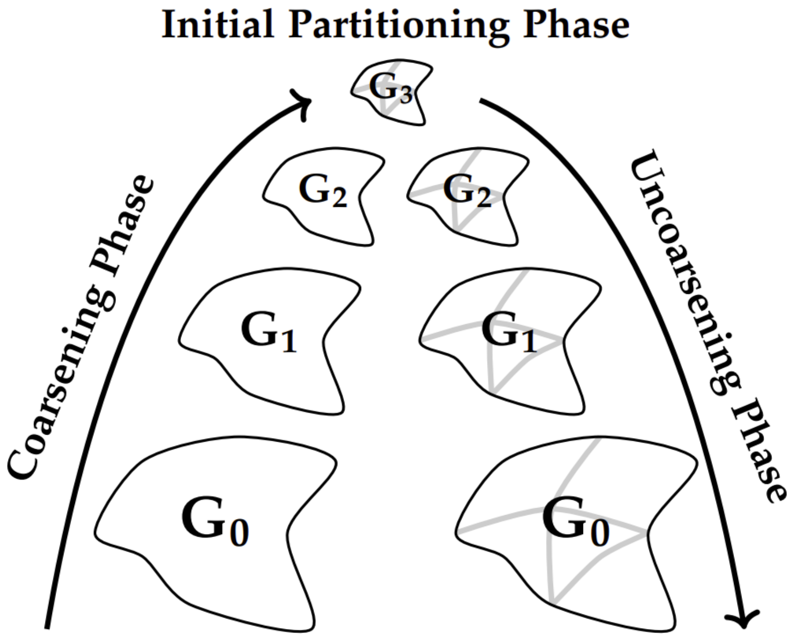


$$w_v: V \to \mathbb{N}$$

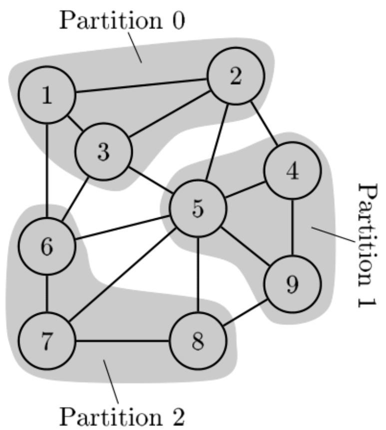
$$w_v(v_i) = 1 \Leftrightarrow v_i \in V_0$$

HNA*: MLkP Phases

- Coarsening Phase:
  recursively collapse nodes into smaller graphs G_{i+1}



- Initial Partitioning Phase:

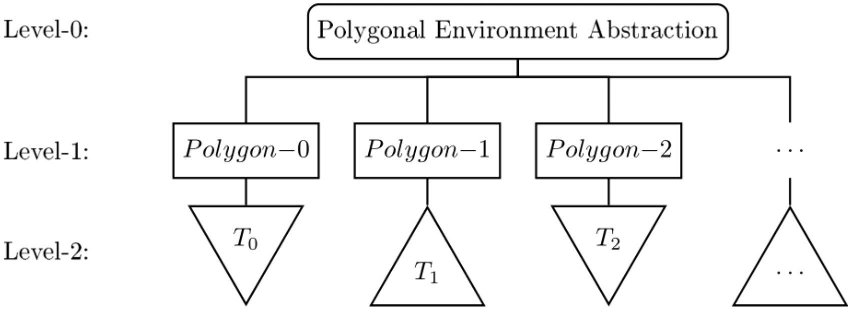


$$w_v: V \to \mathbb{N}$$

$$w_v(v_i) = 1 \Leftrightarrow v_i \in V_0$$

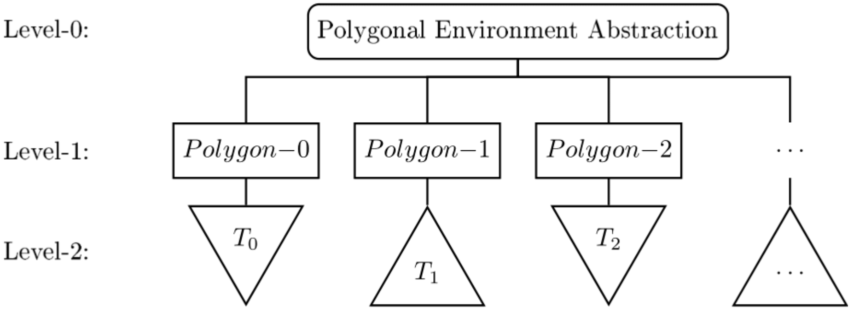
- Uncoarsening Phase:
  iteratively refine the partitioning scheme

MPHGLT2D – Introduction

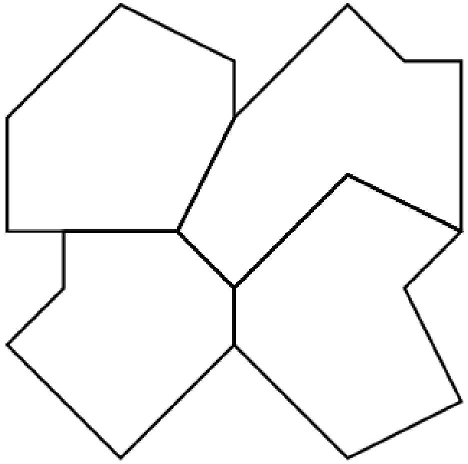


- Depth = Level of detail
- Level-0:
- Level-$i \in (0..n)$:
- Level-$n = 2$:

MPHGLT2D – Introduction

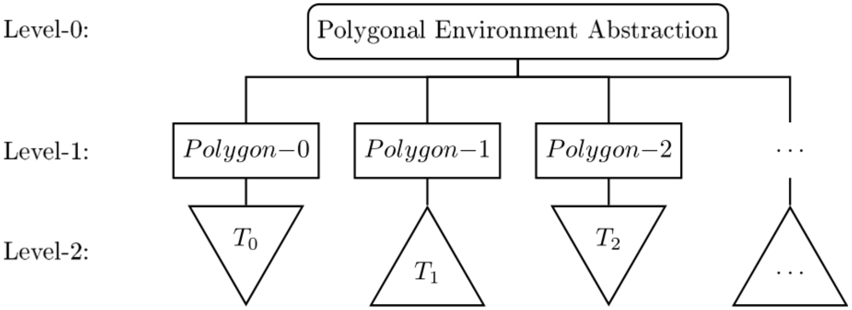


- Depth = Level of detail
- Level-0:
  - Initial polygonal environment description
  - Graph $G_0 = (V_0, E_0), |V_0| = 1, |E_0| = 0$
- Level-$i \in (0..n)$:

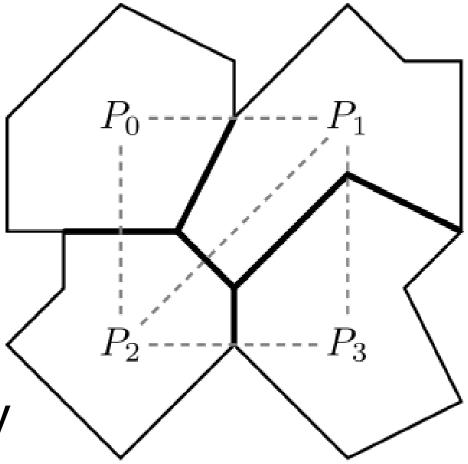


- Level-$n = 2$:

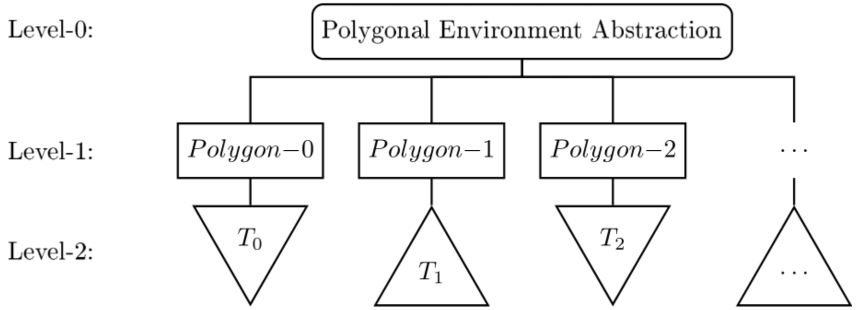
MPHGLT2D – Introduction



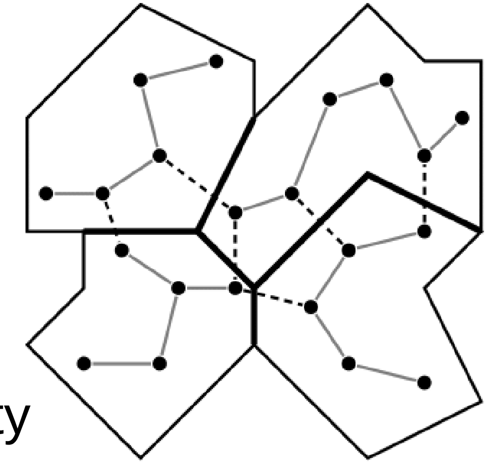
- Depth = Level of detail
- Level-0:
  - Initial polygonal environment description
  - Graph $G_0 = (V_0, E_0), |V_0| = 1, |E_0| = 0$
- Level-$i \in (0..n)$:
  - Intermediate levels: “Regions”
  - Algorithmically deduced (HNA* approach)
  - Explicitly injected by user with N_e-connectivity
- Level-$n = 2$:



MPHGLT2D – Introduction

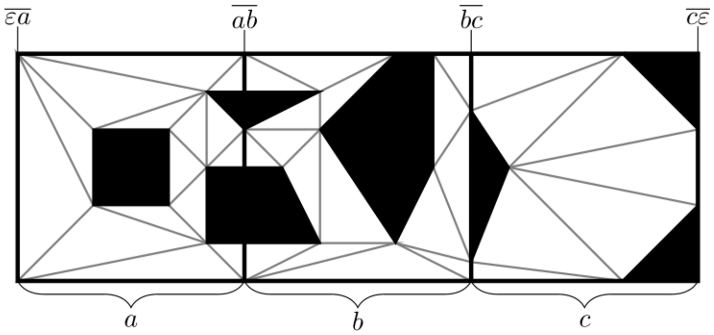


- Depth = Level of detail
- Level-0:
  - Initial polygonal environment description
  - Graph $G_0 = (V_0, E_0), |V_0| = 1, |E_0| = 0$
- Level-$i \in (0..n)$:
  - Intermediate levels: “Regions”
  - Algorithmically deduced (HNA* approach)
  - Explicitly injected by user with N_e-connectivity
- Level-$n = 2$:
  - Parameter n: highest level of detail
  - Triangulation level (CDT)



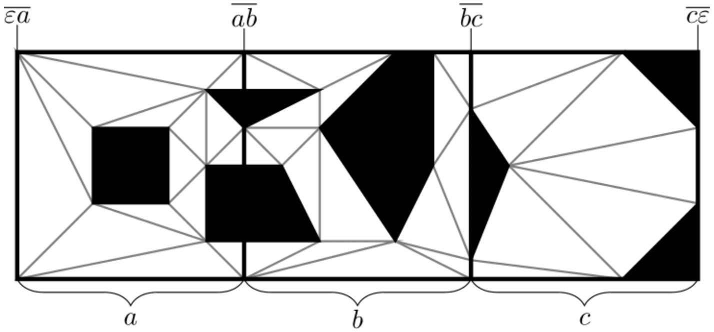
MPHGLT2D – Example

- Example environment (CDT)
  - Square Regions: a, b, c (3×3)
  - Line Borders: $\overline{\varepsilon a} = \overline{a\varepsilon}$ (3)
  - 3 Levels: $[0..2]$

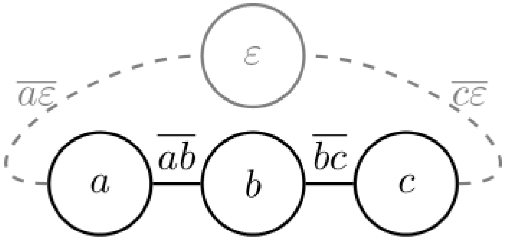


MPHGLT2D – Example

- Example environment (CDT)
  - Square Regions: a, b, c (3×3)
  - Line Borders: $\overline{\varepsilon a} = \overline{a\varepsilon}$ (3)
  - 3 Levels: $[0..2]$



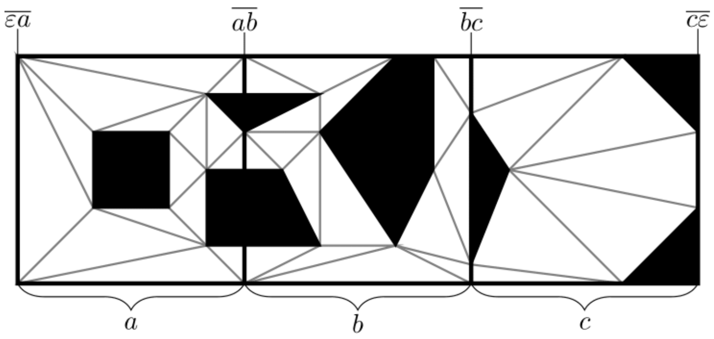
- Search graphs:
  - Level-1 graph



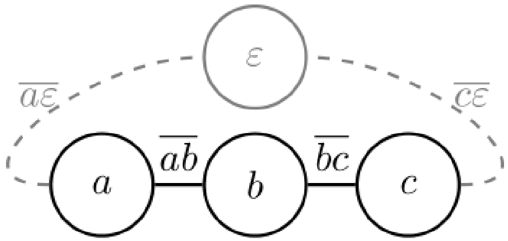
How to search?
Values of $\overline{ab}, \overline{bc}$?

MPHGLT2D – Example

- Example environment (CDT)
  - Square Regions: a, b, c (3×3)
  - Line Borders: $\overline{\varepsilon a} = \overline{a\varepsilon}$ (3)
  - 3 Levels: $[0..2]$

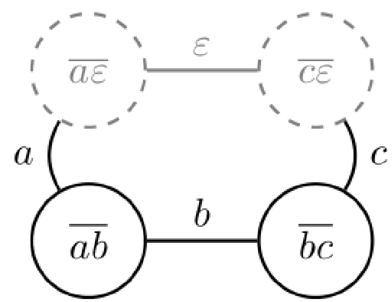


- Search graphs:
  - Level-1 graph



How to search?
Values of $\overline{ab}, \overline{bc}$?

  - Level-1 line graph

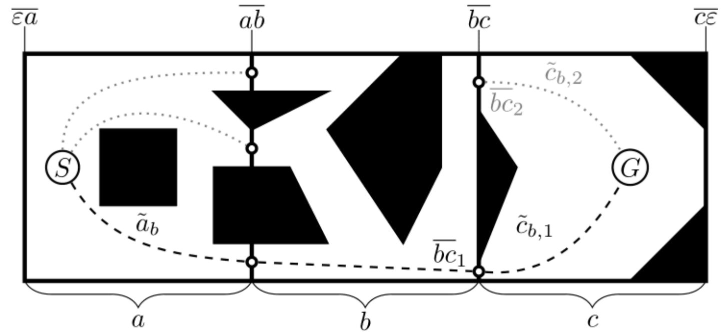


Convert:
Vertex ⟷ Edge

MPHGLT2D – Search

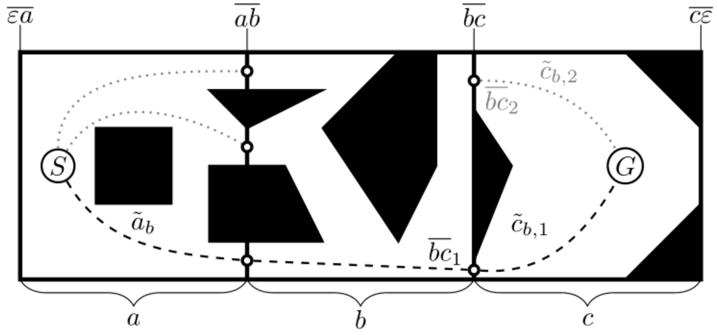
+

- Environment:
  - Start node S
  - Goal node G
  - Gateways $\overline{bc}_j$ of border $\overline{bc}$
  - Sub-region $\tilde{a}_b$ in region a to $\overline{ab}$

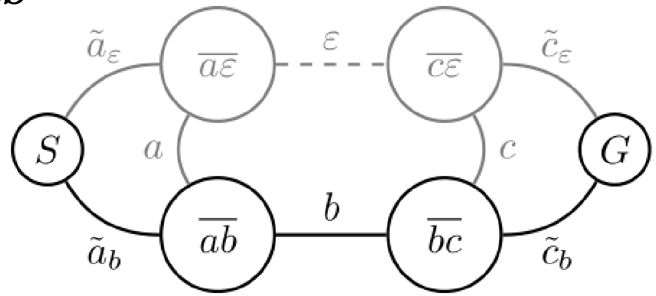


MPHGLT2D – Search

+



- Environment:
  - Start node S
  - Goal node G
  - Gateways $\overline{bc}_j$ of border $\overline{bc}$
  - Sub-region $\tilde{a}_b$ in region a to $\overline{ab}$

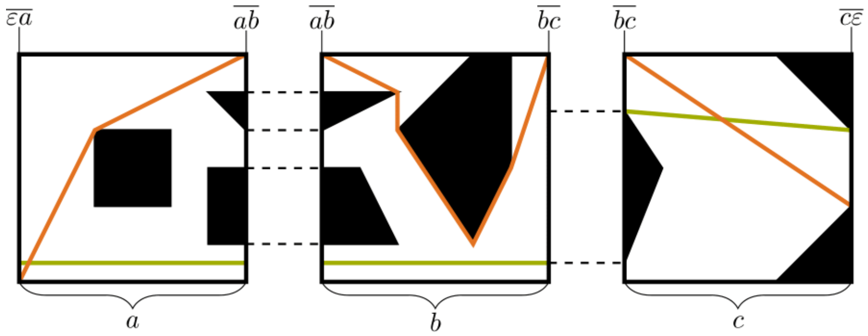


- Level-1 line graph search:
  - Edge weights?
  - Connect & calculate:
    - Start node via $\tilde{a}_{b,i}$: $distance(S, \overline{ab}_i)$
    - Gateways in b: $distance(\overline{ab}_i, \overline{bc}_j)$
    - Goal node via $\tilde{c}_{b,j}$: $distance(\overline{bc}_j, G)$

MPHGLT2D – Region Analysis

+

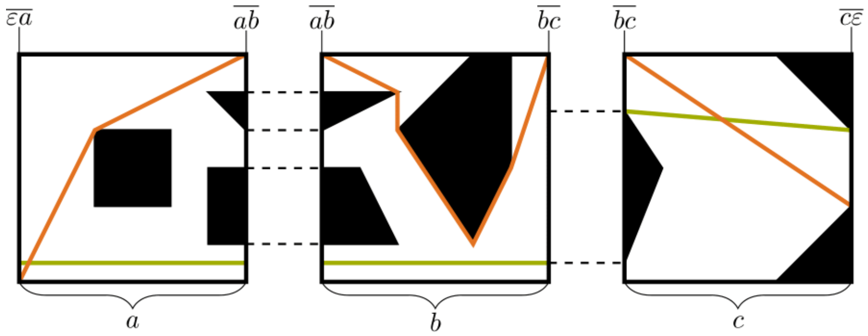
- Region $b = \overline{\overline{ab}\,\overline{bc}}$
- Monte Carlo Simulation:
  - Mean value $\overline{\chi}$ **avg**
  - Standard deviation σ **std**
  - Minimum shortest path **min**
  - Maximum shortest path **max**



MPHGLT2D – Region Analysis

+

- Region $b = \overline{\overline{ab}\;\overline{bc}}$
- Monte Carlo Simulation:
  - Mean value $\bar{\chi}$ **avg**
  - Standard deviation σ **std**
  - Minimum shortest path **min**
  - Maximum shortest path **max**



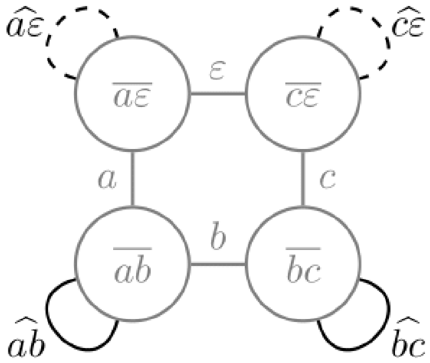
	avg	std	min	max
Square	3.23	**0.26**	3.00	4.24
Region *a*	4.28	**0.44**	3.44	5.50
Region *b*	6.71	**1.84**	3.02	9.57
Region *c*	3.80	**0.13**	3.51	4.08

Uniform distribution along borders

MPHGLT2D – Border Analysis

+

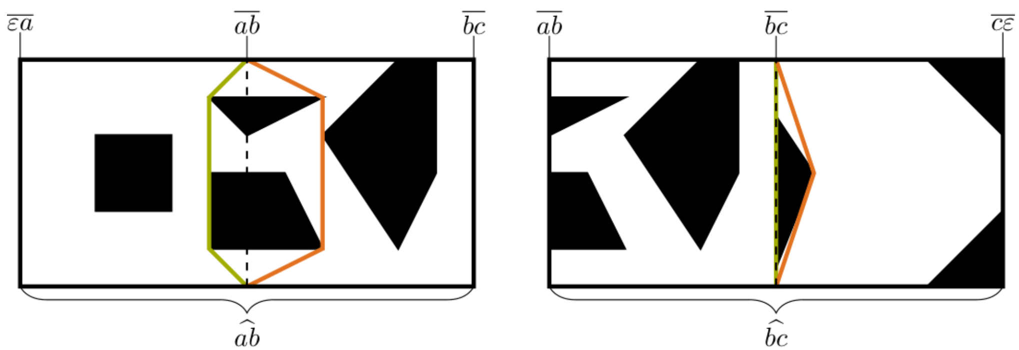
- Reflexive edges $\widehat{a\varepsilon}, \widehat{ab}, \widehat{bc}, \widehat{c\varepsilon}$

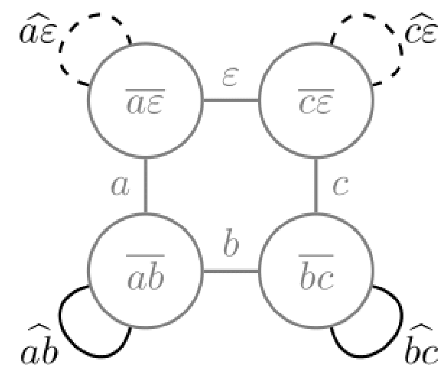


MPHGLT2D – Border Analysis

+

- Reflexive edges $\widehat{a\varepsilon}, \widehat{ab}, \widehat{bc}, \widehat{c\varepsilon}$
- True maximum shortest path
- False maximum shortest path





	avg	std	min	max
Line	1.00	**0.71**	0.00	3.00
Border $\overline{ab}$	2.35	**1.87**	0.33	5.83
Border $\overline{bc}$	1.60	**1.32**	0.33	3.60

MPHGLT2D – Use Cases

“not knowing exact paths”

- Path approximations:

- Dynamic adaptions:

MPHGLT2D – Use Cases

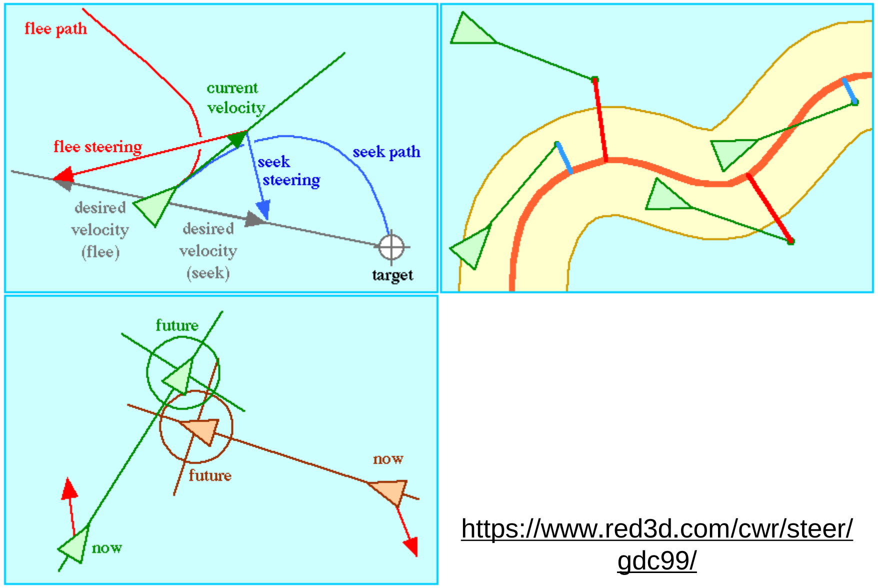
“not knowing exact paths”

- Path approximations:
  - (Pre-)calculate expected paths of all regions
  - Calculate & render exact pathing only for regions on screen
- Dynamic adaptions:

MPHGLT2D – Use Cases

"not knowing exact paths"

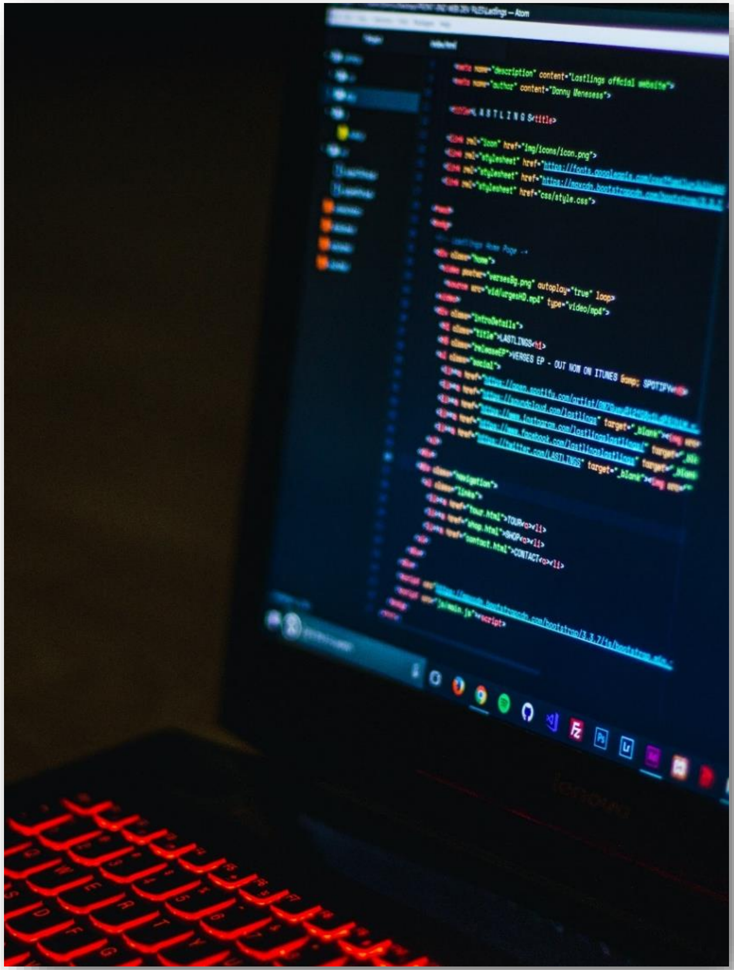
- Path approximations:
  - (Pre-)calculate expected paths of all regions
  - Calculate & render exact pathing only for regions on screen
- Dynamic adaptions:
  - Steering behaviors:
    - Path following
    - Collision avoidance
  - Congestions
  - Etc.



https://www.red3d.com/cwr/steer/gdc99/

Implementation - MPHGLT2D

- C++20
- CDT: CGAL 5.2.3
- A*: Boost 1.75
- IDE: CLion
- Compiler: GCC, clang
- Build: CMAKE 3.20+
- Visualization: Qt5
- Testing: doctest
- Package manager: msys2
- Python3



https://www.pexels.com/photo/photo-of-turned-on-laptop-computer-943096/

Conclusion

- NavMeshes:
  - – Geometrically more complex than grids
  - + Good space abstraction & CDT beneficial search graph
- HPA*:
  - – Grid based
  - + Caching performant & useful for RTS
- HNA*:
  - + NavMesh based
  - – Uses external tools, complex
- MPHGLT2D:
  - + Tries to combine aspects of both approaches
  - – Not yet fully implemented

Conclusion

- NavMeshes:
  – Geometrically more complex than grids
  + Good space abstraction & CDT beneficial search graph
- HPA*:
  – Grid based
  + Caching performant & useful for RTS
- HNA*:
  + NavMesh based
  – Uses external tools, complex
- MPHGLT2D:
  + Tries to combine aspects of both approaches
  – Not yet fully implemented

Hierarchical graphs provide good heuristics

Future Work & Discussion

- Reduce user guidance & implement all desired features
- Make full use of C++20
- Use A* variations: IDA*, TRA*, D*, etc.
- Proper path refinement & self-learning environment
- Parallelize algorithms on GPU via CUDA / OpenCL
- Integrate framework into game prototype

List of References

I. Attribution free images from: https://www.pexels.com/

“Geometric mosaic ornament on tiled floor” – Mathias P.R. Reding: https://www.pexels.com/photo/geometric-mosaic-ornament-on-tiled-floor-4489336/

“Prosthetic Arm on Blue Background” – ThisIsEngineering: https://www.pexels.com/photo/prosthetic-arm-on-blue-background-3913025/

“Photo of Turned on Laptop Computer” – Danny Meneses: https://www.pexels.com/photo/photo-of-turned-on-laptop-computer-943096/

List of References

II. Mentioned software and video games:

1. MassMotion: https://www.oasys-software.com/products/pedestrian-simulation/massmotion/
2. The Lord of the Rings: The Battle for Middle-earth: https://www.ea.com/games/lord-of-the-rings/the-lord-of-the-rings-the-battle-for-middle-earth
3. Sid Meier’s Civilization V: https://civilization.com/civilization-5/
4. METIS: https://github.com/KarypisLab/METIS
5. CUDA: https://developer.nvidia.com/cuda-toolkit
6. OpenCL: https://www.khronos.org/opencl/
7. Days Gone: https://www.bendstudio.com/game/days-gone
8. Age of Darkness: Final Stand: https://www.ageofdarkness.com/

List of References

III. Bibliography:

1. D. Demyen and M. Buro, “Efficient triangulation-based pathfinding,” in Proceedings of The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA, 2006, pp. 942–947.
2. D. Demyen and M. Buro, “Efficient triangulation-based pathfinding,” in Proceedings of The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA, 2006, pp. 942–947.
3. S. J. Russel and P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed. Prentice Hall, 2010, isbn: 9781292153964.
4. J.-P. Laumond, Robot Motion Planning and Control. Springer, 1998, isbn: 9783540762195.

List of References

5. Y. Björnsson and K. Halldórsson, “Improved Heuristics for Optimal Path-finding on Game Maps,” in Proceedings of the Second Artificial Intelligence and Interactive Digital Entertainment Conference, J. E. Laird and J. Schaeffer, Eds., AAAI Press, 2006, pp. 9–14.
6. M. Pinter. “Toward More Realistic Pathfinding.” (2001), [Online]. Available: https://www.gamedeveloper.com/programming/toward-more-realistic-pathfinding (visited on 10/11/2021).
7. A. Patel. “Grid pathfinding optimizations.” (2020), [Online]. Available: https://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html#heuristicsfor-grid-maps (visited on 10/11/2021).
8. K. G. Shin and P. Ramanathan, “RealTime Computing: A New Discipline of Computer Science and Engineering,” in Proceedings of IEEE, Special Issue on RealTime Systems, vol. 82, 1994.

List of References

9. L. R. M. Gonzalez, “Sketching for Real-time Control of Crowd Simulations,” Ph.D. dissertation, University of Sheffield, 2021.
10. M. Jansen and N. Sturtevant, “Direction Maps for Cooperative Pathfinding,” Proceedings of the 4th Artificial Intelligence and Interactive Digital Entertainment Conference, AIIDE 2008, pp. 185–190, 2008.
11. C. W. Reynolds, “Flocks, Herds and Schools: A Distributed Behavioral Model,” SIGGRAPH Comput. Graph., vol. 21, no. 4, pp. 25–34, Aug. 1987, issn: 0097-8930. doi: 10.1145/37402.37406.
12. C. W. Reynolds, “Steering Behaviors For Autonomous Characters,” in Proceedings of Game Developers Conference 1999 held in San Jose, California. Miller Freeman Game Group, San Francisco, California, 1999, pp. 763–782.
13. H. S. M. Coxeter, Regular Polytopes. Courier Corporation, 1973, isbn: 9780486614809.

List of References

14. P. Tozour, “AI Game Programming Wisdom,” in S. Rabin, Ed. Charles River Media, 2002, ch. Building a Near-Optimal Navigation Mesh, pp. 171–185, isbn: 9781584500773.
15. P. Schorn and F. Fisher, “Graphics gems IV,” in P. S. Heckbert, Ed. AP Professional, 1994, ch. I.2 Testing the convexity of a polygon, pp. 7–15, isbn: 9780123361554. doi: 10.1016/C2013-0-07360-4.
16. G.-J. Giezeman and W. Wesselink, “2D polygons,” in CGAL User and Reference Manual, 5.2.3, CGAL Editorial Board, 2021.
17. D. Brewer, “Tactical Pathfinding on a NavMesh,” in Game AI Pro 360, S. Rabin, Ed., Boca Raton: CRC Press, 2019, pp. 361–368, isbn: 9780429054969. doi: 10.1201/ 9780429054969.
18. M. Yvinec, “2D triangulation,” in CGAL User and Reference Manual, 5.2.3, CGAL Editorial Board, 2021.

List of References

19. M. Kallmann, “Path Planning in Triangulations,” in Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games, Jan. 2005, pp. 49– 54.
20. N. Pelechano and C. Fuentes, “Hierarchical Path-Finding for Navigation Meshes (HNA*),” Computers & Graphics, vol. 59, pp. 68–78, 2016, issn: 0097-8493. doi: 10.1016/j.cag.2016.05.023.
21. R. Oliva and N. Pelechano, “NEOGEN: Near Optimal Generator of Navigation Meshes for 3D Multi-Layered Environments,” Computers & Graphics, pp. 1–11, Apr. 2013. doi: 10.1016/j.cag.2013.03.004.
22. A. Botea, M. Müller, and J. Schaeffer, “Near Optimal Hierarchical Path-Finding (HPA*),” Journal of Game Development, vol. 1, pp. 1–30, Jan. 2004.

List of References

23. A. Kring, A. J. Champandard, and N. Samarin, “DHPA* and SHPA*: Efficient Hierarchical Pathfinding in Dynamic and Static Game Worlds,” in Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AAAI Press, 2010, pp. 39–44.
24. H. Samet, “An Overview of Quadtrees, Octrees, and Related Hierarchical Data Structures,” in Theoretical Foundations of Computer Graphics and CAD, R. A. Earnshaw, Ed., Springer Berlin Heidelberg, 1988, pp. 51–68, isbn: 978-3-642-83539-1. doi: 10.1007/978-3-642-83539-1_2.
25. A. Yahja, A. Stentz, S. Singh, and B. Brumitt, “Framed-quadtree path planning for mobile robots operating in sparse environments,” in Proceedings of IEEE Conference on Robotics and Automation (ICRA), vol. 1, Jun. 1998, pp. 650–655, isbn: 0-7803-4300- X. doi: 10.1109/ROBOT.1998.677046.

List of References

26. G. Karypis and V. Kumar, “Multilevel k-way Partitioning Scheme for Irregular Graphs,” Journal of Parallel and Distributed Computing, vol. 48, no. 1, pp. 96–129, 1998, issn: 0743-7315. doi: https://doi.org/10.1006/jpdc.1997.1404.
27. B. W. Kernighan and S. Lin, “An efficient heuristic procedure for partitioning graphs,” The Bell System Technical Journal, vol. 49, no. 2, pp. 291–307, 1970. doi: 10.1002/j.1538-7305.1970.tb01770.x.