Reasoning with Ontologies for Non-Player Character Decision-Making in Games

DC - AIIDE 2022, Cal Poly Pomona, USA

PhD Student: Sylvain LAPEYRADE
Supervisors: Christophe REY, Bruno BACHELET, Loïc YON

LIMOS – Clermont Auvergne University, France

26 October 2022

PhD context

Context: Game designers want **possible** “emergent behaviours” for their NPCs

PhD context

Context: Game designers want **possible** "emergent behaviours" for their NPCs

What are emergent behaviours?

A behaviour that is not explicitly developed by the game designer

PhD context

Context: Game designers want **possible** "emergent behaviours" for their NPCs

What are emergent behaviours?

A behaviour that is not explicitly developed by the game designer

Why possible?

Emergent behaviours are not **always** desirable:

- Game designers may want to **fully control** player experience
- Emergent behaviours are more likely to be **incoherent**
- Players might want to be able to **predict** NPCs behaviour

Emergent behaviours

Why designers would want emergent behaviours anyway?

- To avoid having to anticipate every possible case
- To enable a more personalised experience for the player
- To be less predictible for the player

Game AI behavioural techniques state of the art[1]

- Ad hoc algorithms (Hard-coded, poor re-usability)
- Finite-State Machines (Hard-coded, poor scaling)
- Behavioural Trees (Hard-coded, poor scaling)
- Utility-based AI (Tuning utilities can be laborious)
- Action Planning (Difficult to use, can be expensive)
- Learning-based AI (Resource intensive, black box)

[1] From reference books in Game AI: Yannakakis and Togelius [2018], Millington [2019] and review article: Simonov et al. [2019]

Is the trend changing?

Observation: Logic-based methods are almost absent from SOTA

Is the trend changing?

Observation: Logic-based methods are almost absent from SOTA

Recent projects using logic-based methods

- **Versu** – Evans and Short [2014]
- **MKULTRA** – Horswill [2015]
- **EmbASP** – Calimeri et al. [2018]
- **UnityIIS** – Brännström and Nieves [2021]
- **VEsNA** – Gatti and Mascardi [2022]
- **ThinkEngine** – Angilica et al. [2022]

Is the trend changing?

Observation: Logic-based methods are almost absent from SOTA

Recent projects using logic-based methods

- **Versu** – Evans and Short [2014]
- **MKULTRA** – Horswill [2015]
- **EmbASP** – Calimeri et al. [2018]
- **UnityIIS** – Brännström and Nieves [2021]
- **VEsNA** – Gatti and Mascardi [2022]
- **ThinkEngine** – Angilica et al. [2022]

Note: They are not yet adopted by the general industry

PhD proposition

Rule-based systems principles to create emergent behaviours

Rule-based systems principles to create emergent behaviours

Declare facts and rules about the game

- Bob is a man.
- A man is a human.
- Any human can move up, down, left and right.
- Any human can shoot an arrow up, down, left and right.

Rule-based systems principles to create emergent behaviours

Declare facts and rules about the game

- Bob is a man.
- A man is a human.
- Any human can move up, down, left and right.
- Any human can shoot an arrow up, down, left and right.

Listing 3: The same predicates in Prolog

```
1 man(bob).
2 human(X):- man(X).
3 move(X, Direction):- human(X),
4     member(Direction, [up, down, right, left]).
5 shoot(X, Direction):- human(X),
6     member(Direction, [up, down, right, left]).
```

Ontologies and knowledge inference

Observation: it seems very intuitive to create rules about a game declaratively, as is done in board games

Observation: it seems very intuitive to create rules about a game declaratively, as is done in board games

Whithout stating it explicitly, it can be infered that:

- Bob is a human
- Bob can move up, down, left and right
- Bob can shoot an arrow up, down, left and right

Observation: it seems very intuitive to create rules about a game declaratively, as is done in board games

Whithout stating it explicitly, it can be infered that:

- Bob is a human
- Bob can move up, down, left and right
- Bob can shoot an arrow up, down, left and right

Conclusion: By simply stating what is true in the world, behaviours not explicitly designed can emerge dynamically

$\Rightarrow$ This means having **planning** and **explicability** is possible!

Limitation to be addressed

Why game designers wanting emergent behaviour do not already use similar techniques?

Limitation to be addressed

Why game designers wanting emergent behaviour do not already use similar techniques?

Game developers are used to imperative programming not declarative programming

Fix: Library or a plugin directly available from game engines

Limitation to be addressed

Why game designers wanting emergent behaviour do not already use similar techniques?

Game developers are used to imperative programming not declarative programming

Fix: Library or a plugin directly available from game engines

Using and making many good ontologies and rules can be difficult

Fix: Establish an efficient methodology for their design

Limitation to be addressed

Why game designers wanting emergent behaviour do not already use similar techniques?

Game developers are used to imperative programming not declarative programming

Fix: Library or a plugin directly available from game engines

Using and making many good ontologies and rules can be difficult

Fix: Establish an efficient methodology for their design

Resources available for NPCs AI can be very limited

Fix: The whole behaviour generation process must be optimised, e.g. with rule pruning, multi-threading, etc.

How to make inferences? Use inference engine like Prolog

We chose SWI-Prolog beacause:

- It's popular and maintained
- It supports the Well-Founed Semantics
- It should be "easier" to interface with game engines

How to make inferences? Use inference engine like Prolog

We chose SWI-Prolog beacause:

- It's popular and maintained
- It supports the Well-Founed Semantics
- It should be "easier" to interface with game engines

Well-Founded Semantics (WFS) – Van Gelder et al. [1991]

- What is unknown is no longer assumed to be *false* (NAF)
- In the WFS a proposition can be *true*, *false* or *undefined*
- *Undefined* values are used for uncertainty and contradiction

Interface between Prolog & game engines

How to make inference engine and game engine interact?

Interface between Prolog & game engines

How to make inference engine and game engine interact?

Connecting Prolog to Unity seems like a recurring endeavour:

- **SwiPICs** – "Official" SWI-Prolog interface, *last updated 2013*
  `github.com/SWI-Prolog/contrib-swiplcs`
- **UnityProlog** – Ian Horswill, *last updated 2017*
  `github.com/ianhorswill/UnityProlog`
- **BackTraQ** – *"Prolog like", last updated 2019*
  `github.com/FacticiusVir/BacktraQ`
- **Yield Prolog** – *"Prolog like", last updated 2019*
  `sourceforge.net/projects/yieldprolog`
- **CSharpProlog** – J.Pool & J.Sakamoto, *last updated 2020*
  `github.com/jsakamoto/CSharpProlog`
- **Pengines.Client** – *F# alternative, last updated 2021(!)*
  `github.com/ninjarobot/Pengines.Client`

Interface between Prolog & game engines

How to make inference engine and game engine interact?

Connecting Prolog to Unity seems like a recurring endeavour:

- **SwiPICs** – "Official" SWI-Prolog interface, *last updated 2013*
  `github.com/SWI-Prolog/contrib-swiplcs`
- **UnityProlog** – Ian Horswill, *last updated 2017*
  `github.com/ianhorswill/UnityProlog`
- **BackTraQ** – *"Prolog like", last updated 2019*
  `github.com/FacticiusVir/BacktraQ`
- **Yield Prolog** – *"Prolog like", last updated 2019*
  `sourceforge.net/projects/yieldprolog`
- **CSharpProlog** – J.Pool & J.Sakamoto, *last updated 2020*
  `github.com/jsakamoto/CSharpProlog`
- **Pengines.Client** – *F# alternative, last updated 2021(!)*
  `github.com/ninjarobot/Pengines.Client`

No entirely satisfying solution...

Interface between Prolog & game engines

How to make inference engine and game engine interact?

Connecting Prolog to Unity seems like a recurring endeavour:

- **SwiPICs** – "Official" SWI-Prolog interface, *last updated 2013*
  `github.com/SWI-Prolog/contrib-swiplcs`
- **UnityProlog** – Ian Horswill, *last updated 2017*
  `github.com/ianhorswill/UnityProlog`
- **BackTraQ** – *"Prolog like", last updated 2019*
  `github.com/FacticiusVir/BacktraQ`
- **Yield Prolog** – *"Prolog like", last updated 2019*
  `sourceforge.net/projects/yieldprolog`
- **CSharpProlog** – J.Pool & J.Sakamoto, *last updated 2020*
  `github.com/jsakamoto/CSharpProlog`
- **Pengines.Client** – *F# alternative, last updated 2021(!)*
  `github.com/ninjarobot/Pengines.Client`

No entirely satisfying solution... so we coded our own interface:
`github.com/sylvainlapeyrade/mqi_csharp`

Global architecture

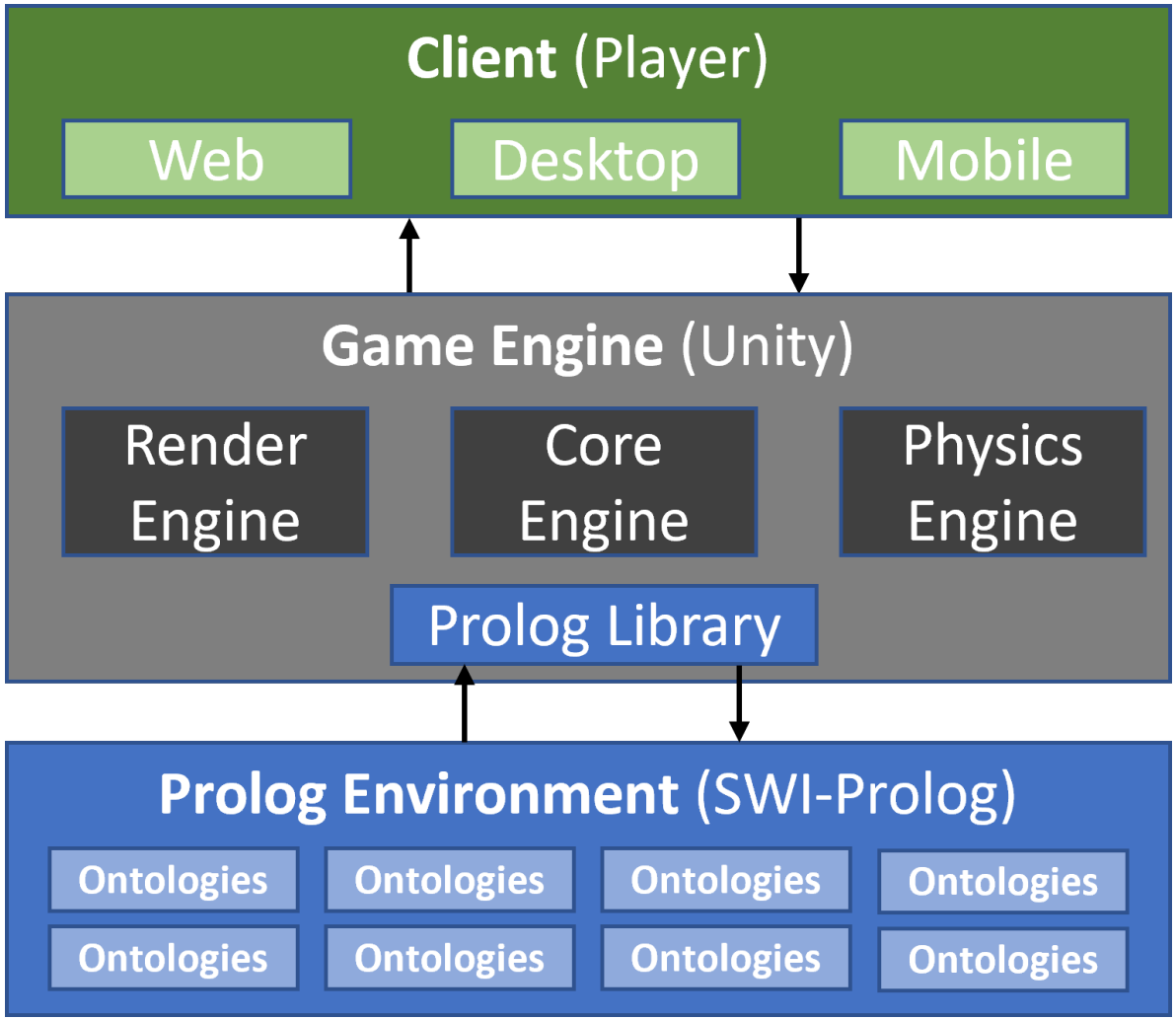
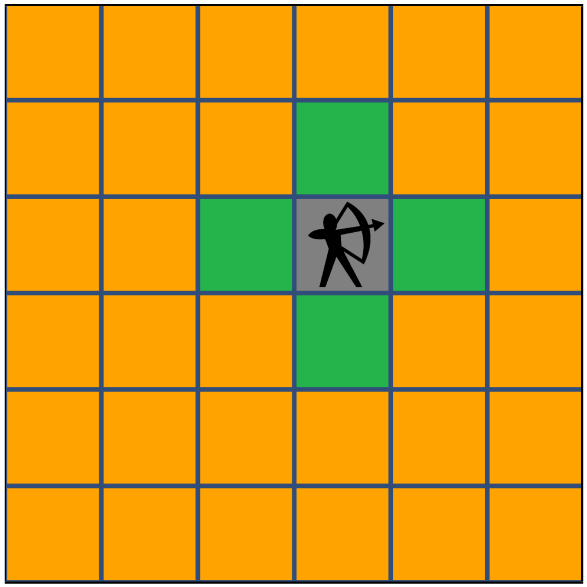
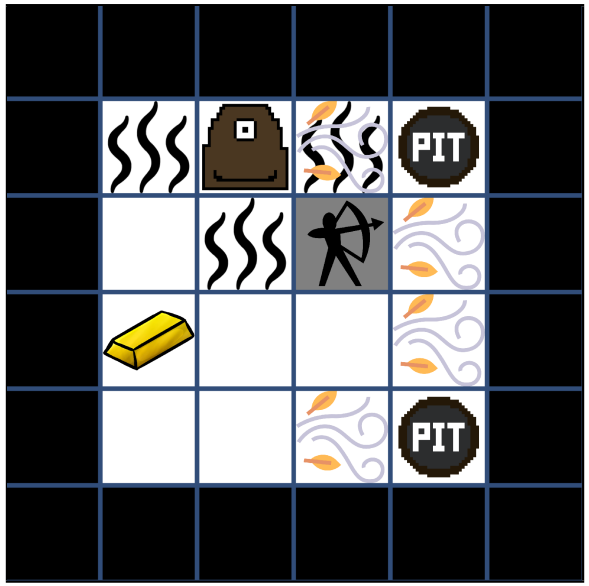


Figure 1: Architecture of the integration of a logic programming environment in a game engine

Prototype: Wumpus World



(a) View of the Agent



(b) View of the world

Principle of Wumpus World

An agent explores a cave, finds the gold and leaves without dying.

Example extended from Russell and Norvig [2021], and Warren [1999].

Prototype: Wumpus World example

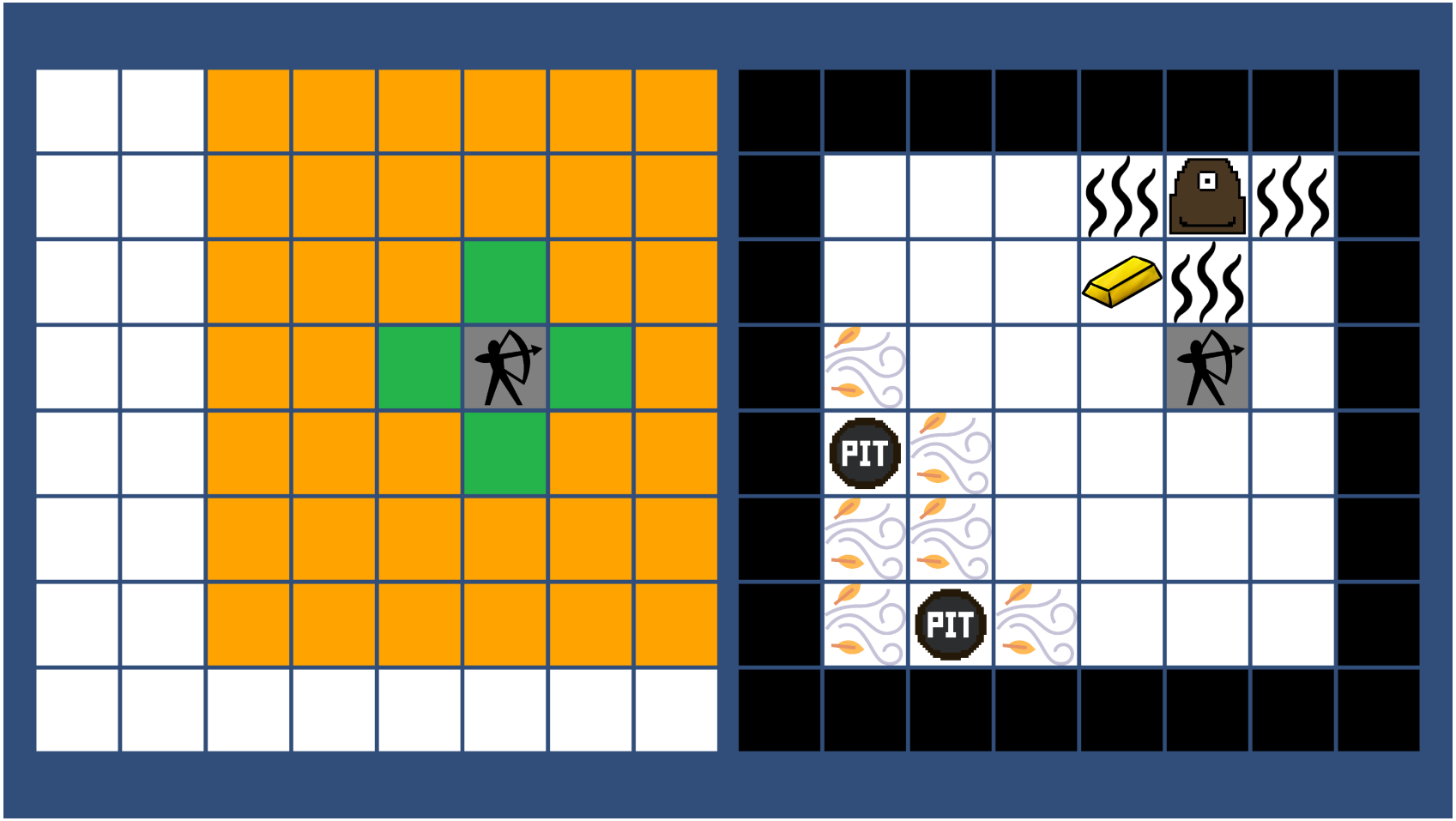


Figure 3: Turn 1.

Prototype: Wumpus World example

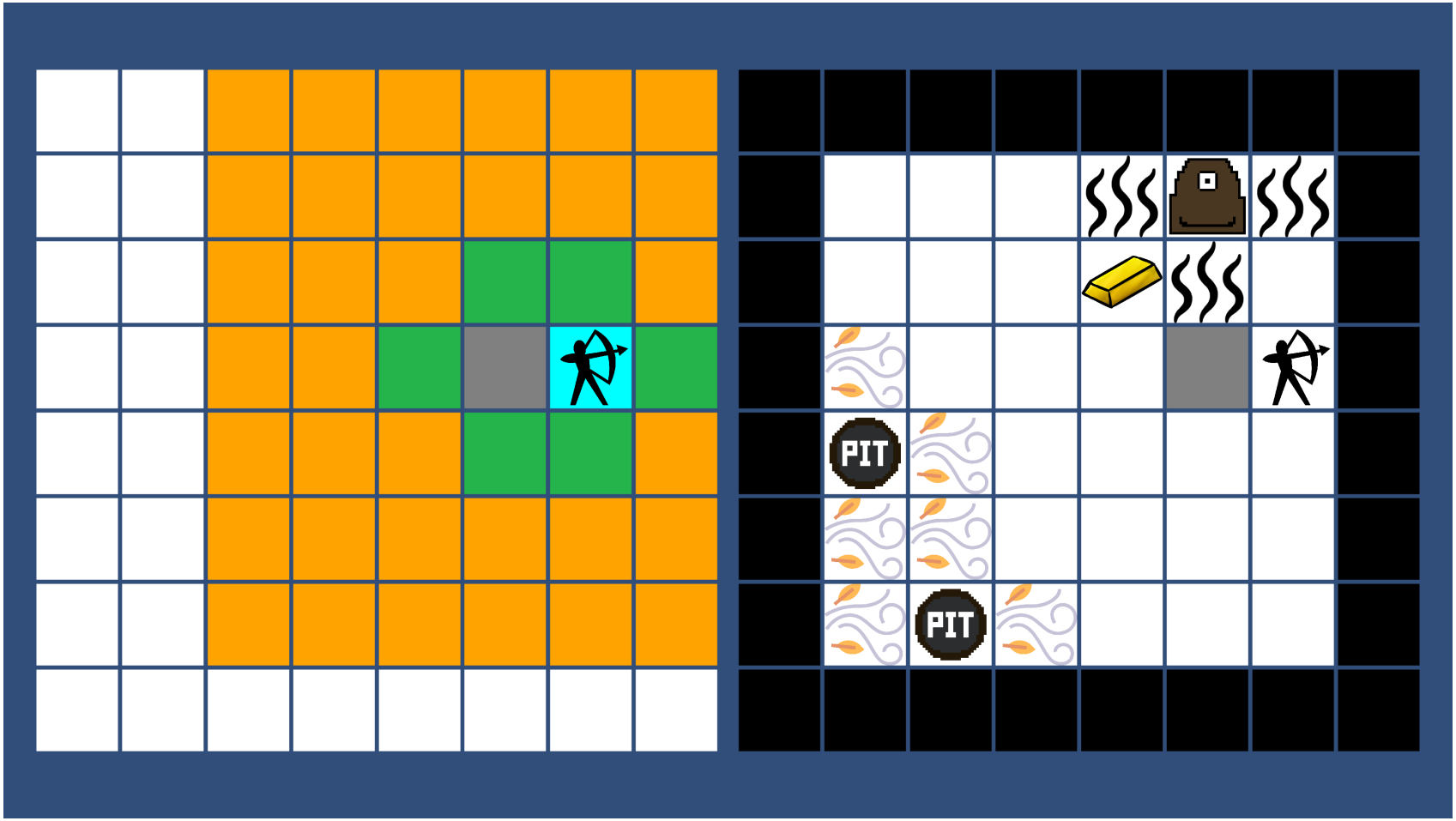


Figure 3: Turn 2.

Prototype: Wumpus World example

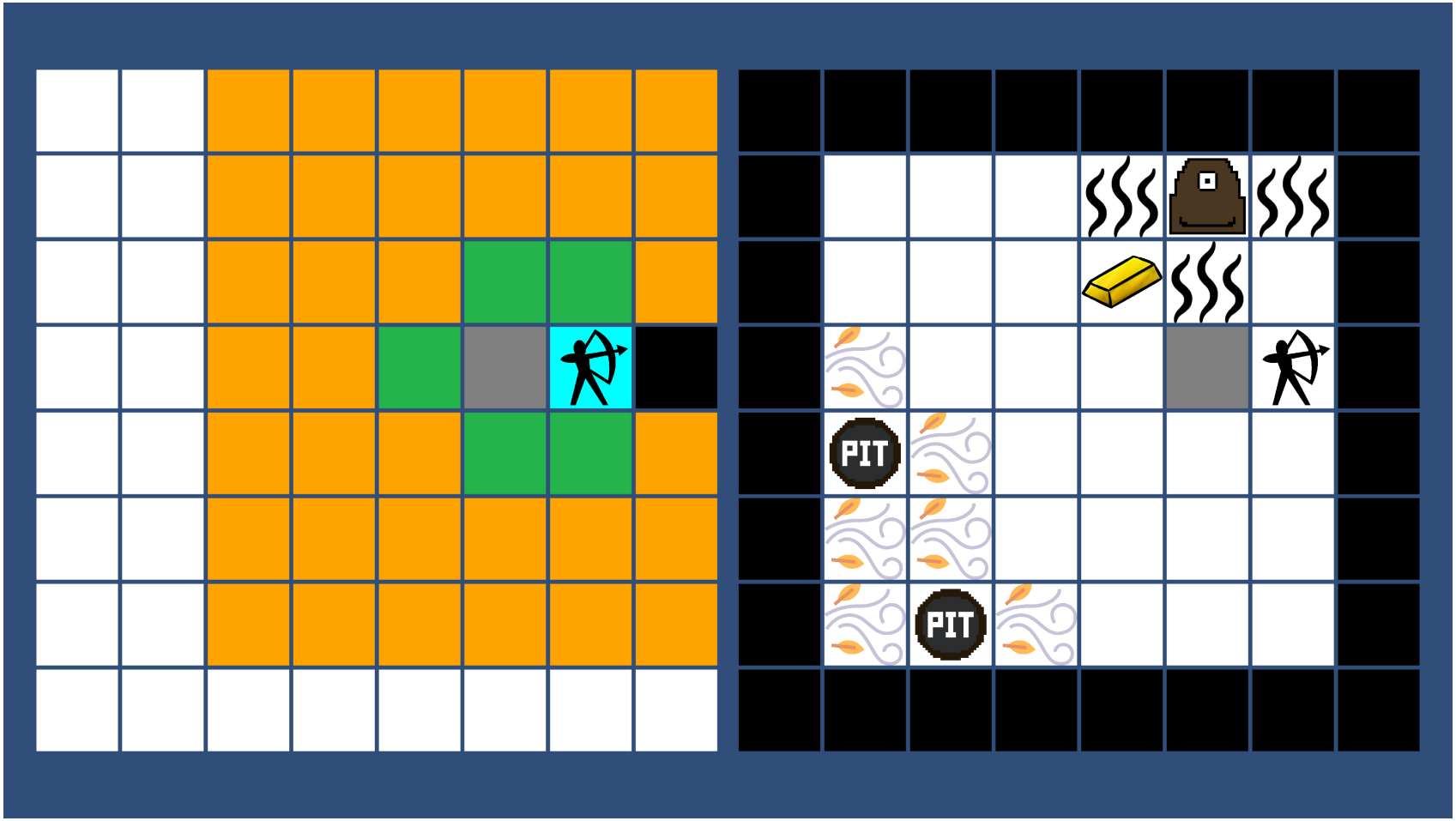


Figure 3: Turn 3.

Prototype: Wumpus World example

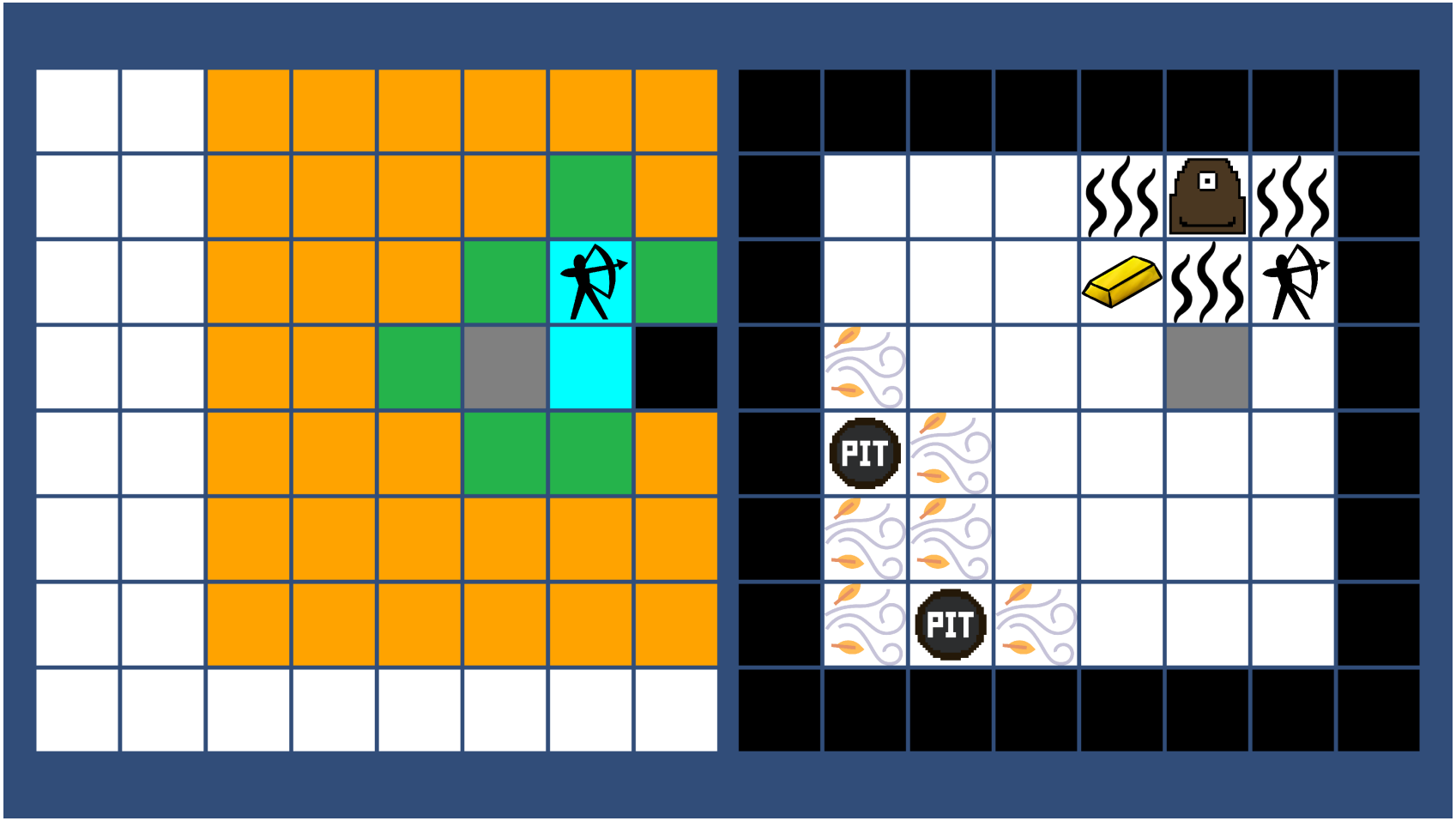


Figure 3: Turn 4.

Prototype: Wumpus World example

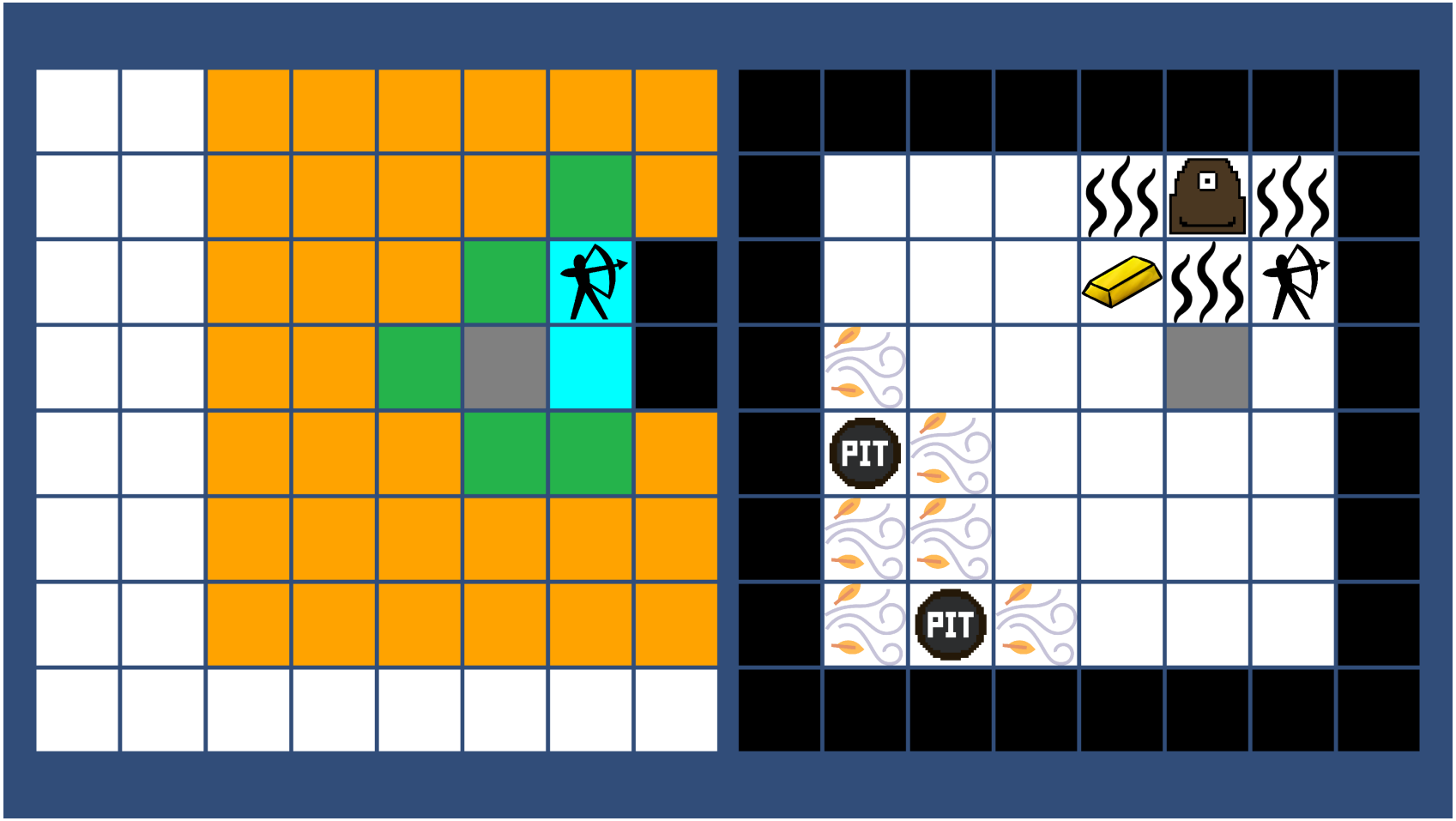


Figure 3: Turn 5.

Prototype: Wumpus World example

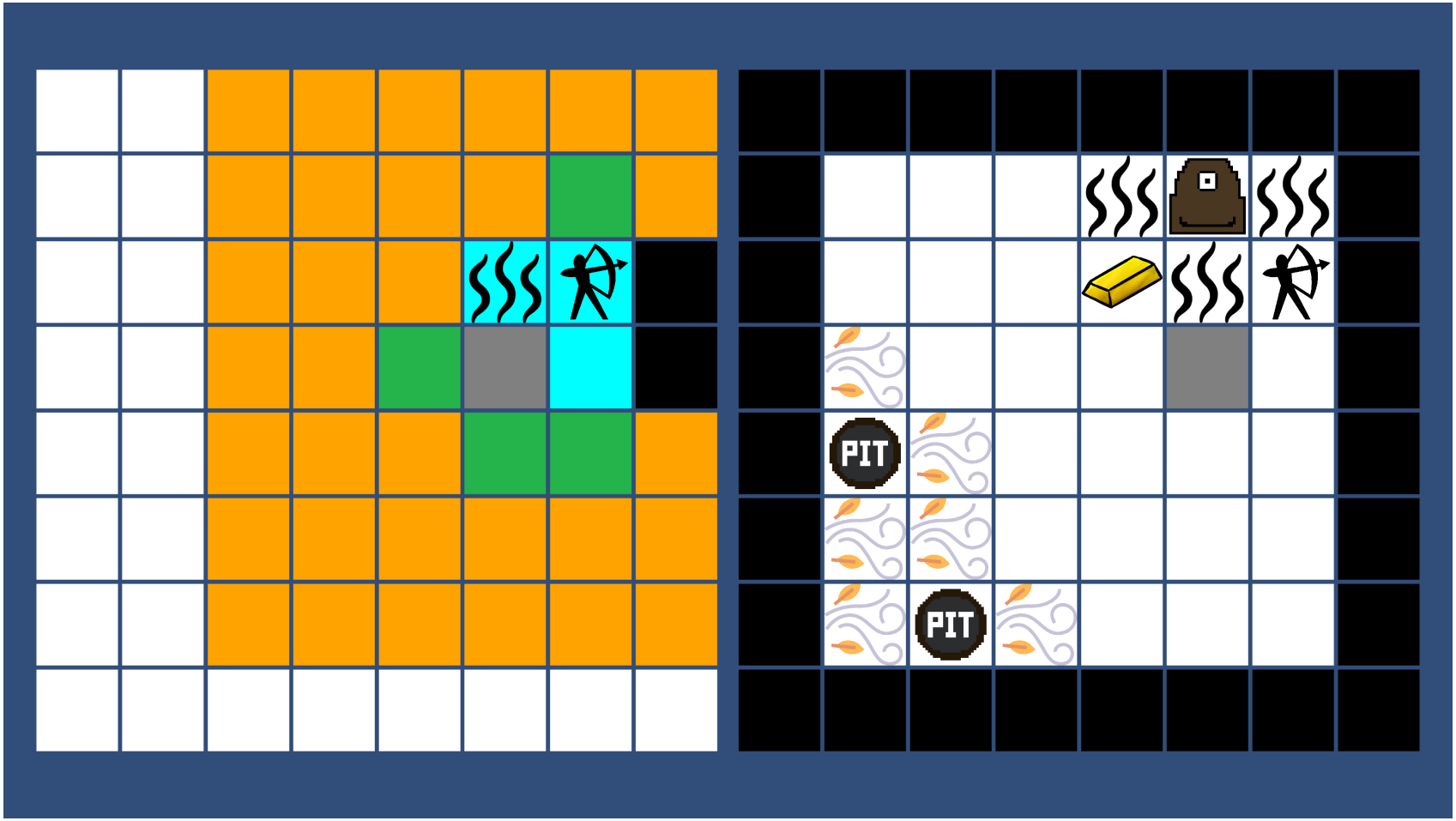


Figure 3: Turn 6.

Prototype: Wumpus World example

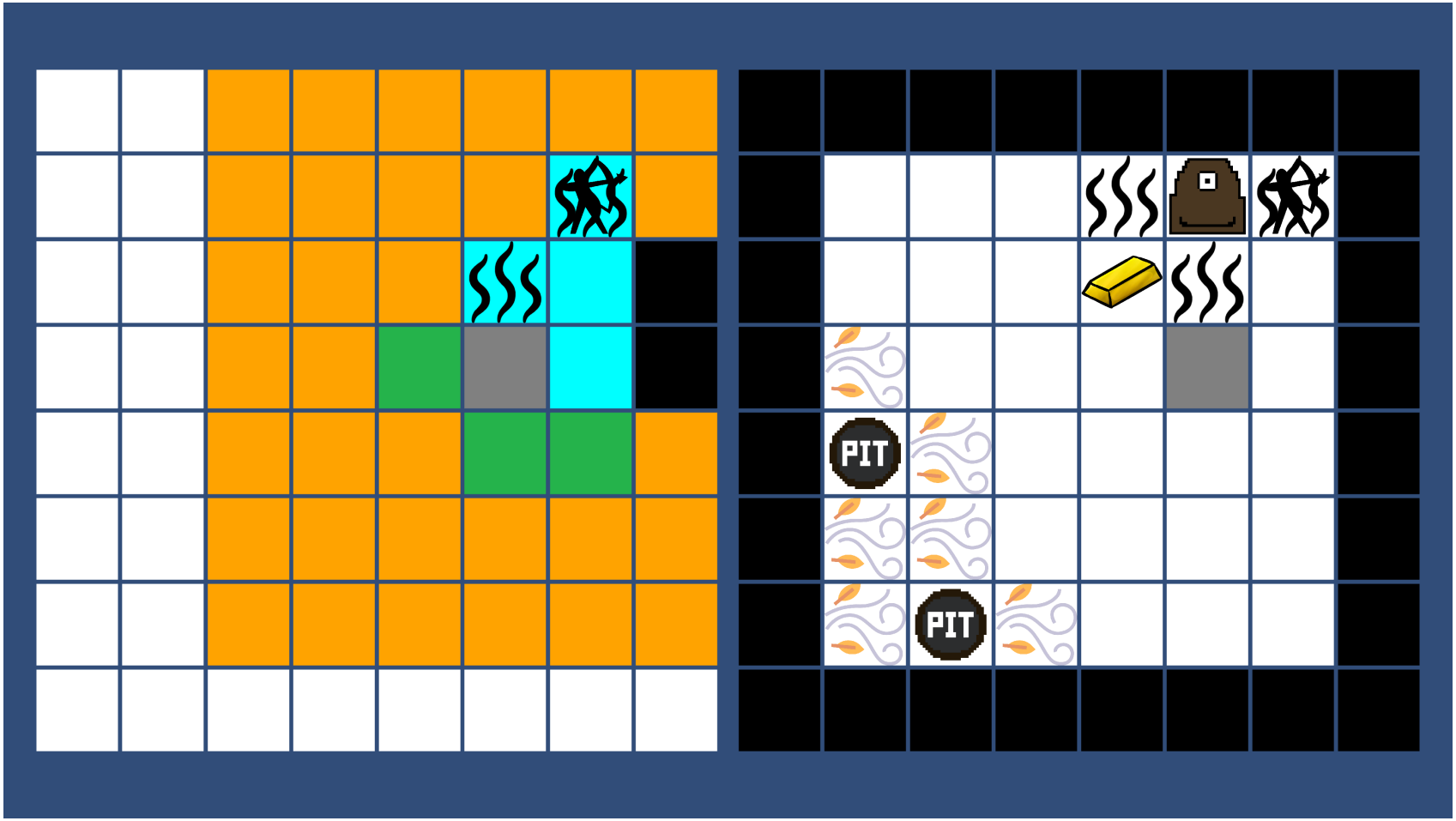


Figure 3: Turn 7.

Prototype: Wumpus World example

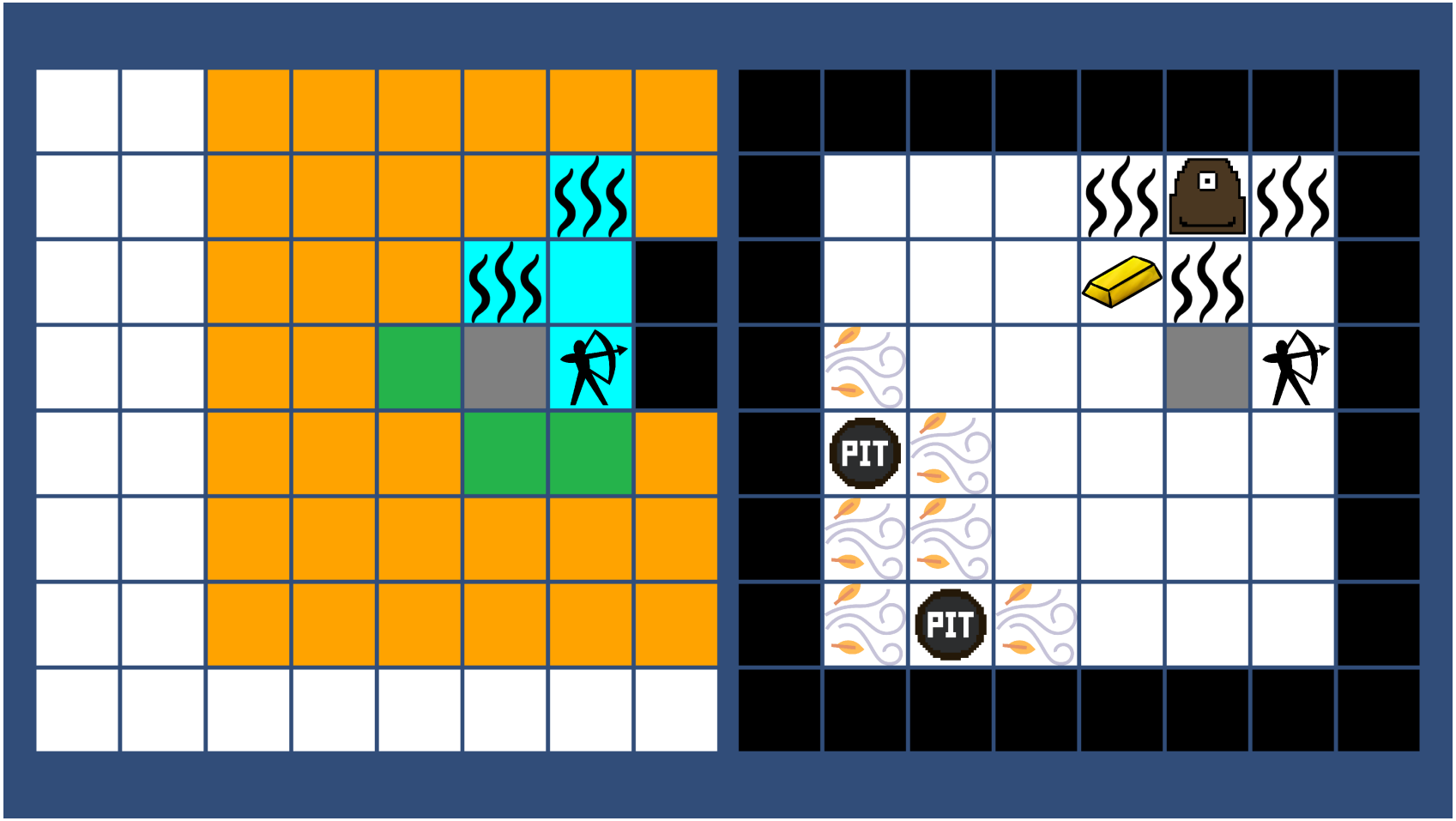


Figure 3: Turn 8.

Prototype: Wumpus World example

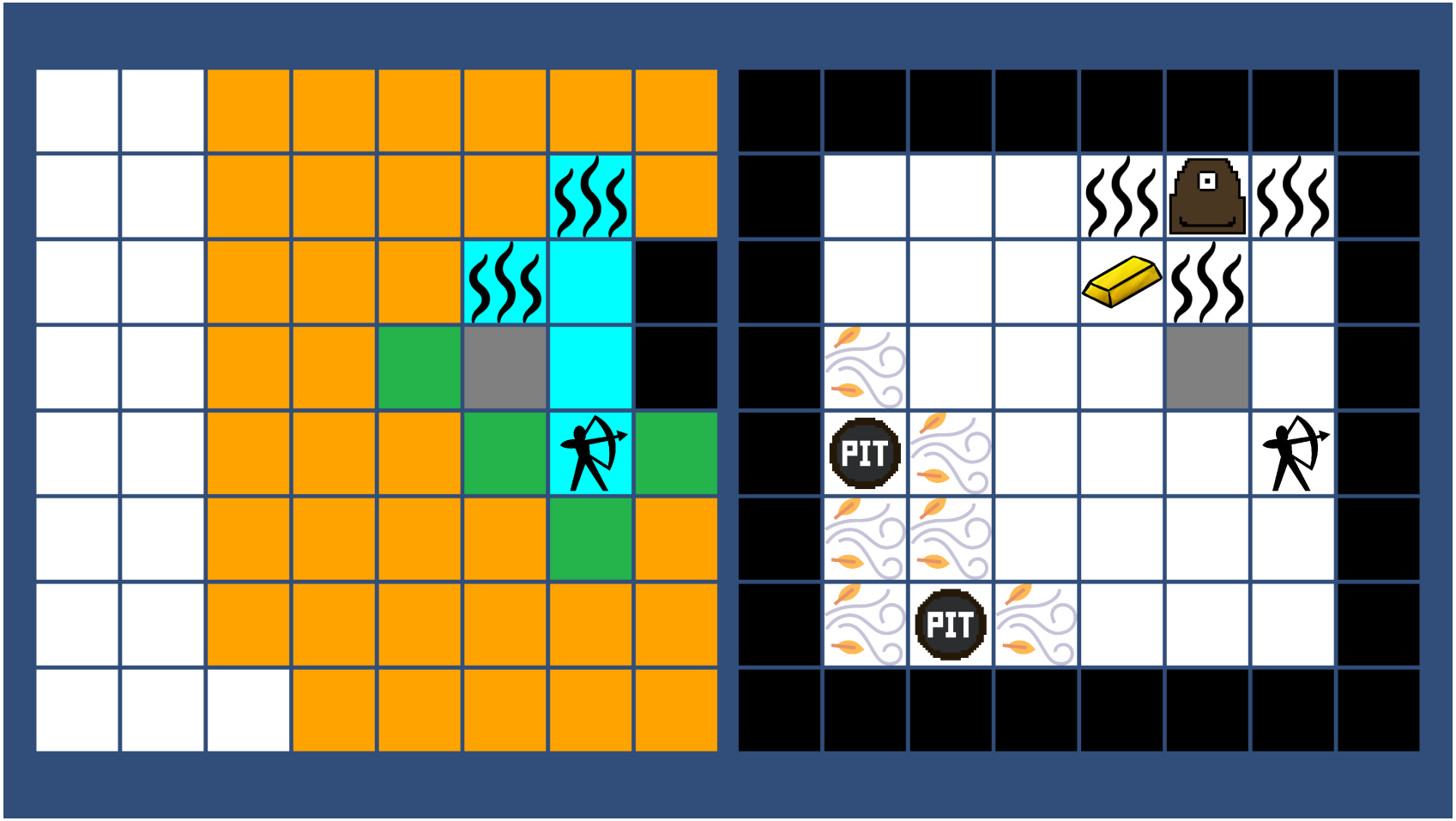


Figure 3: Turn 9.

Prototype: Wumpus World example

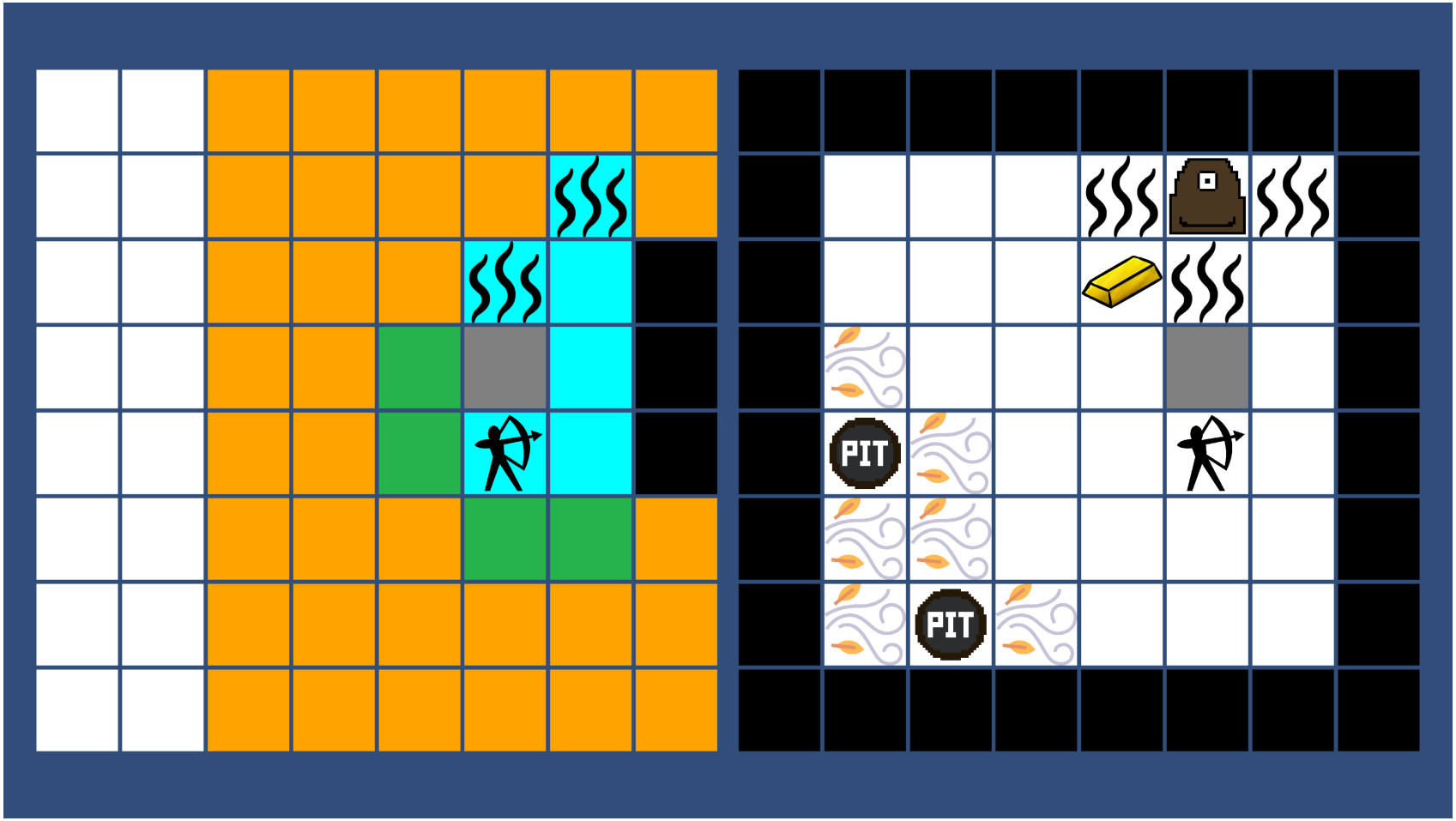


Figure 3: Turn 10.

Prototype: Wumpus World example

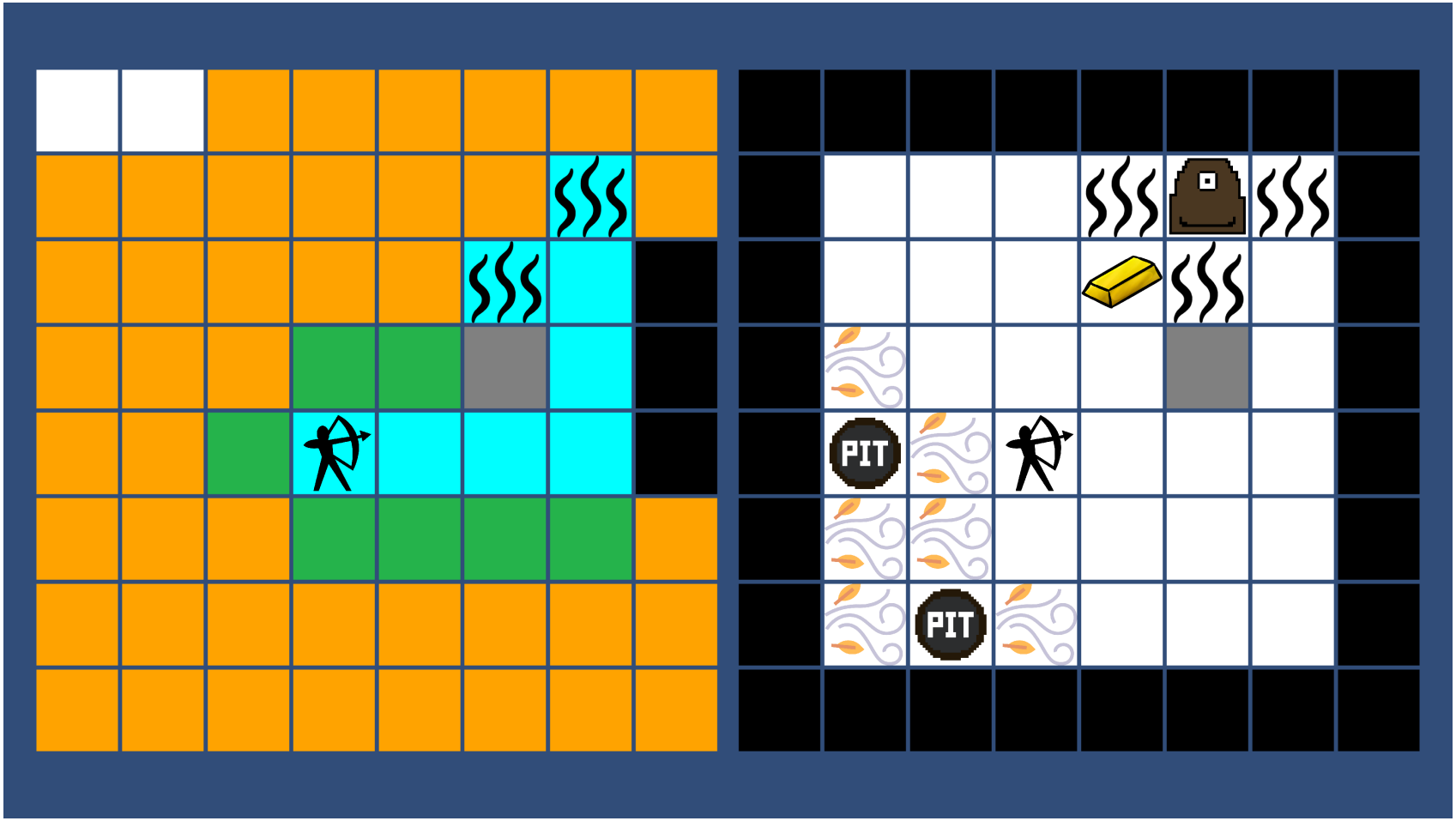


Figure 3: Turn 11.

Prototype: Wumpus World example

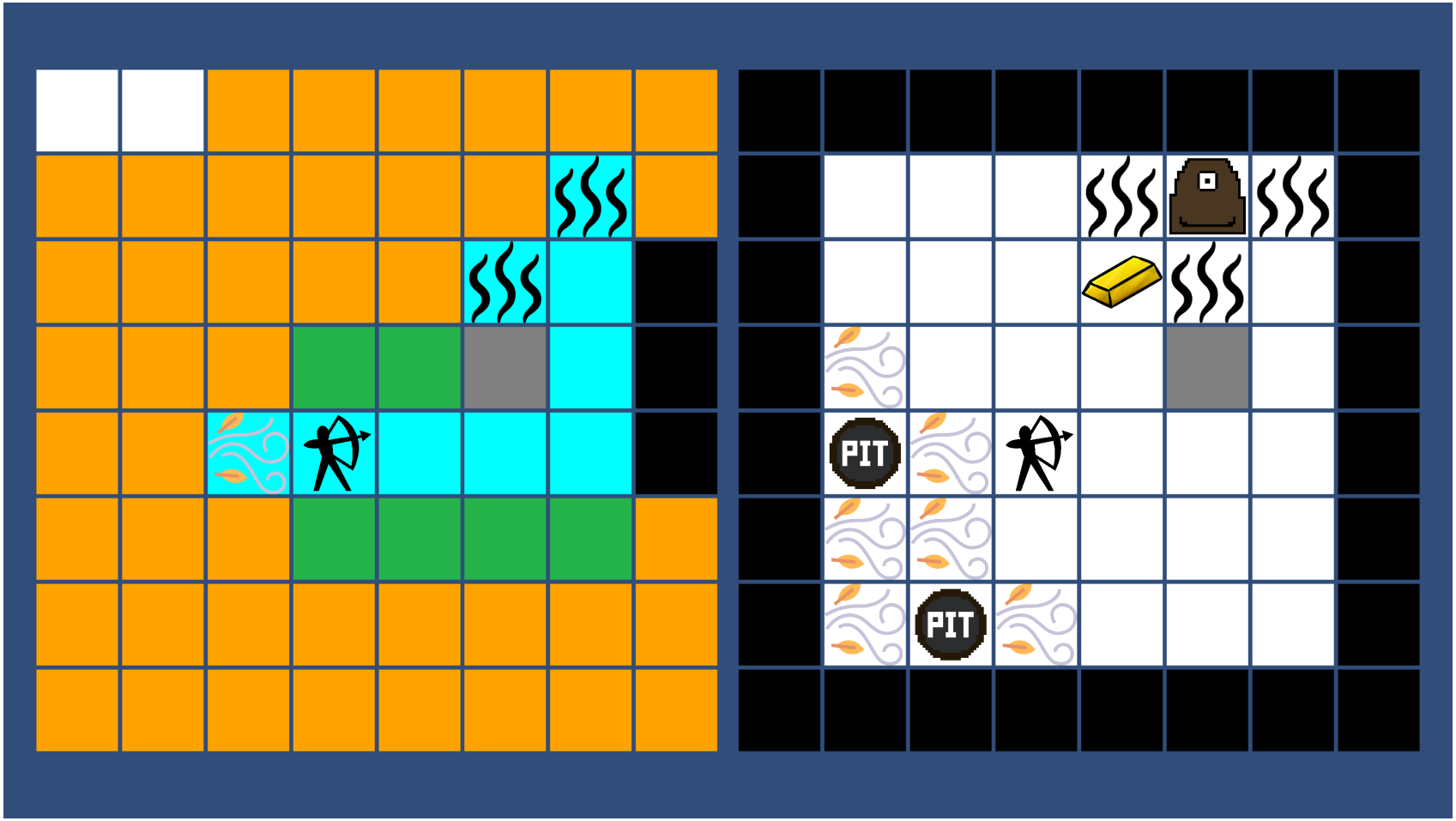


Figure 3: Turn 12.

Prototype: Wumpus World example

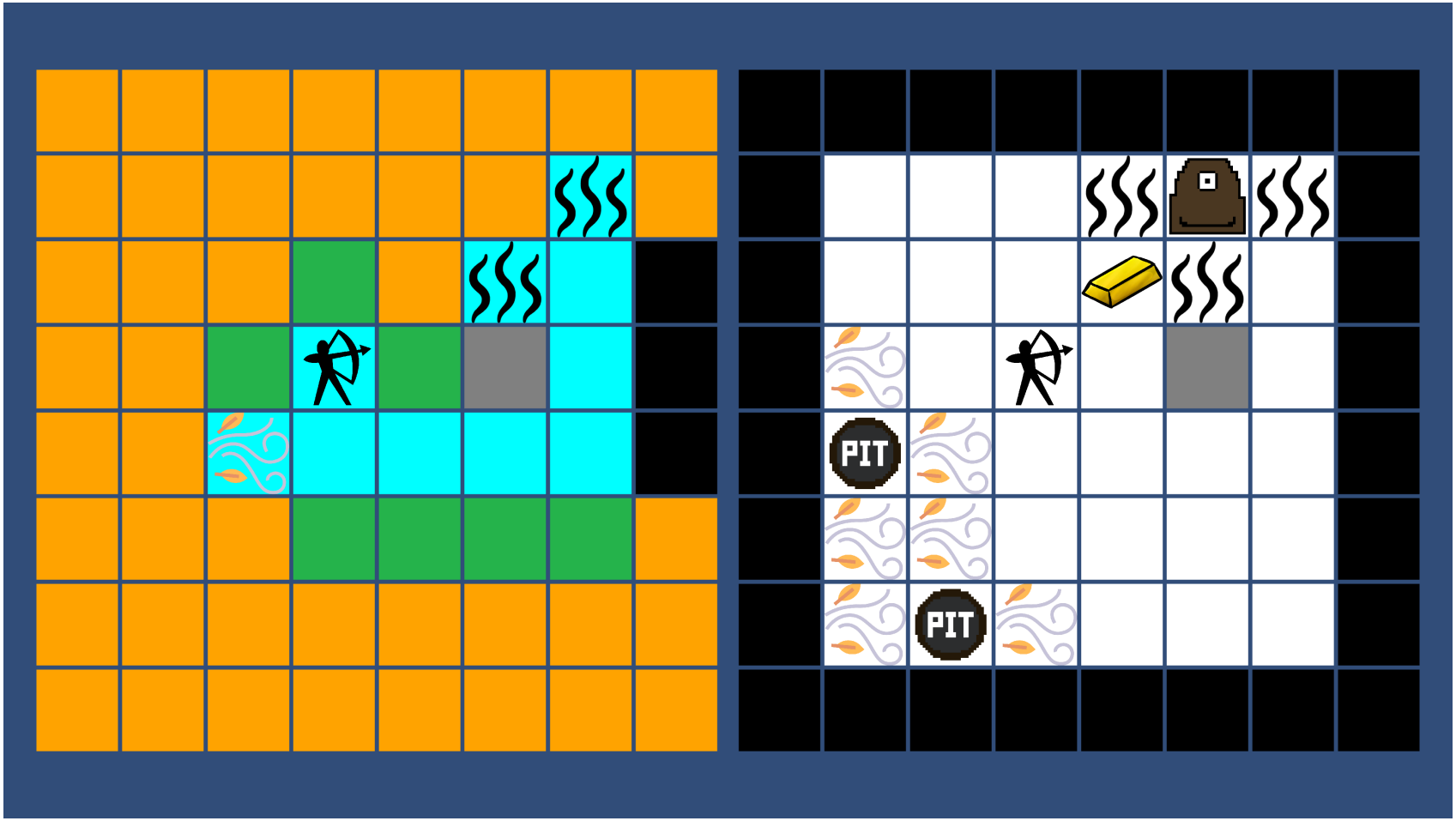


Figure 3: Turn 13.

Prototype: Wumpus World example

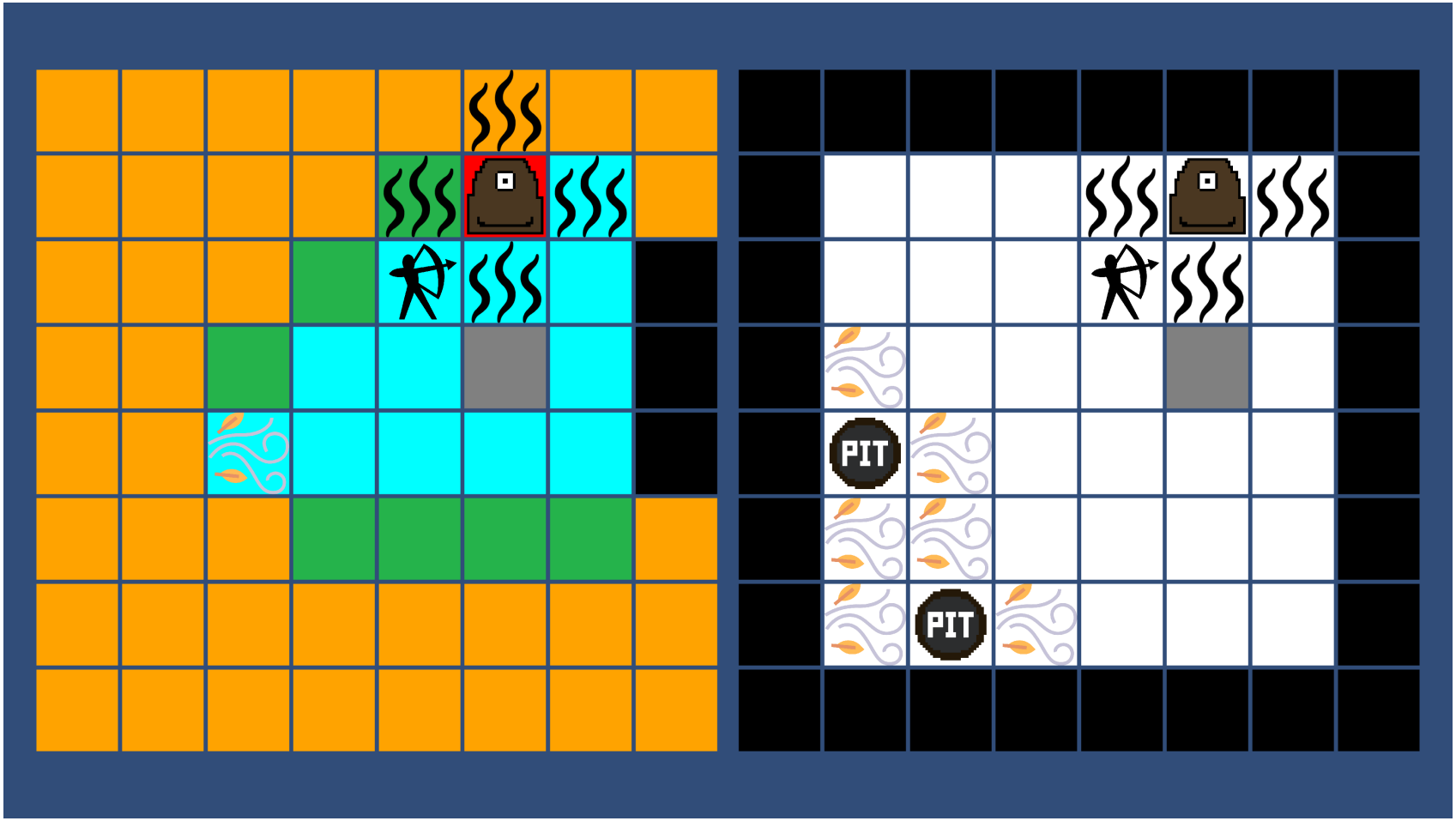


Figure 3: Turn 14.

Prototype: Wumpus World example

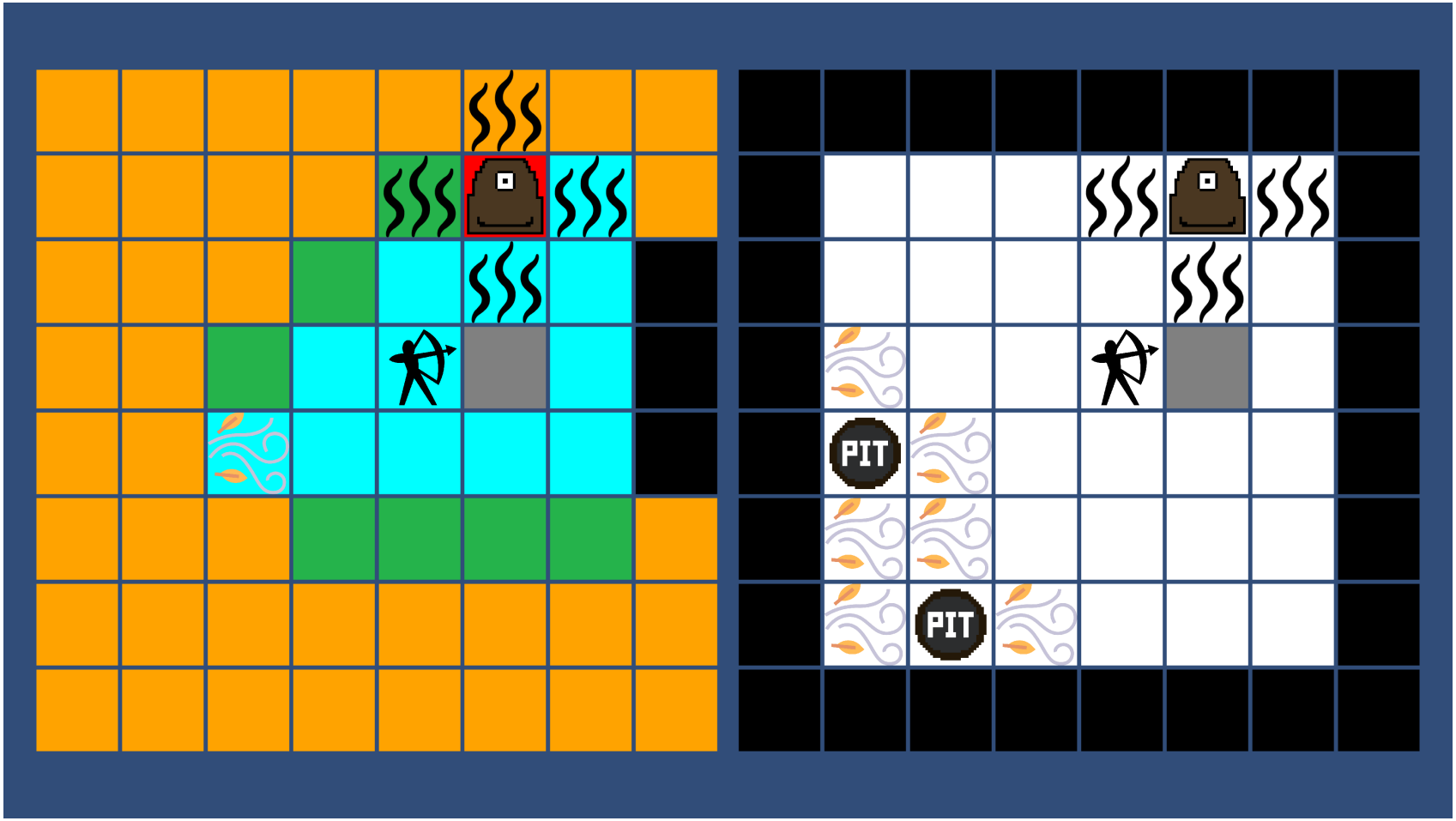


Figure 3: Turn 15.

Prototype: Wumpus World example

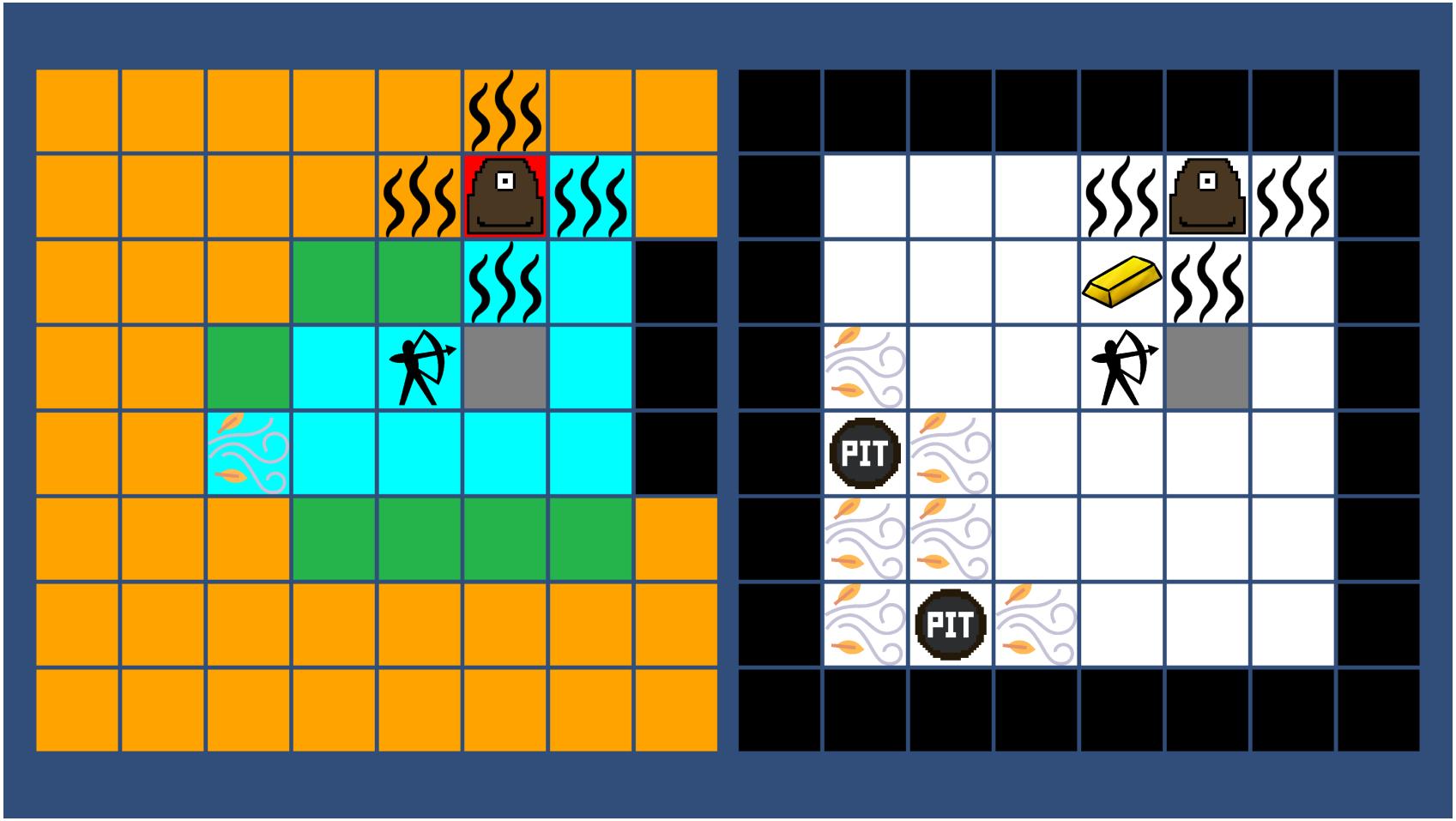


Figure 3: Turn 16.

Prototype: Wumpus World example

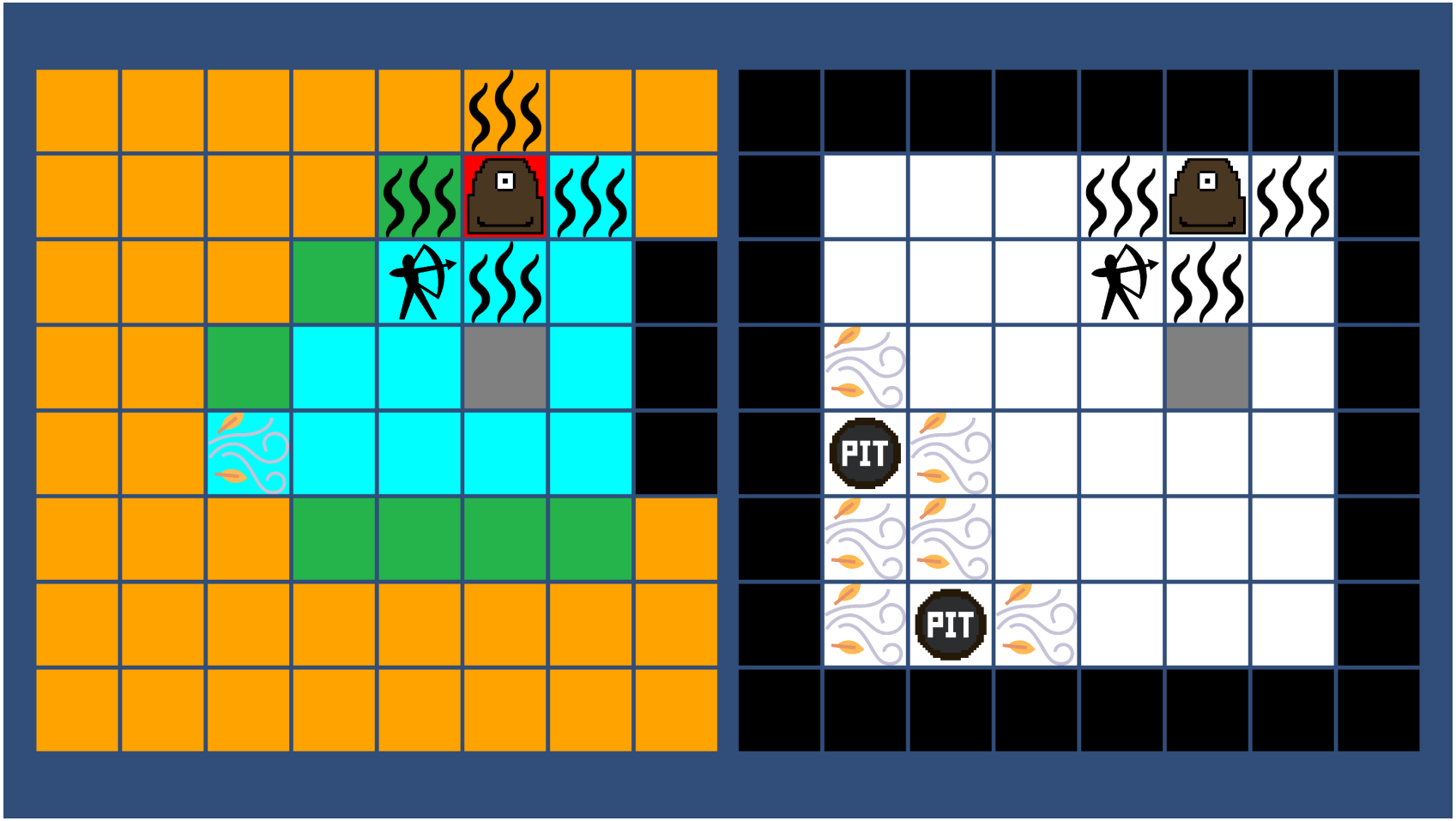


Figure 3: Turn 17.

Prototype: Wumpus World example

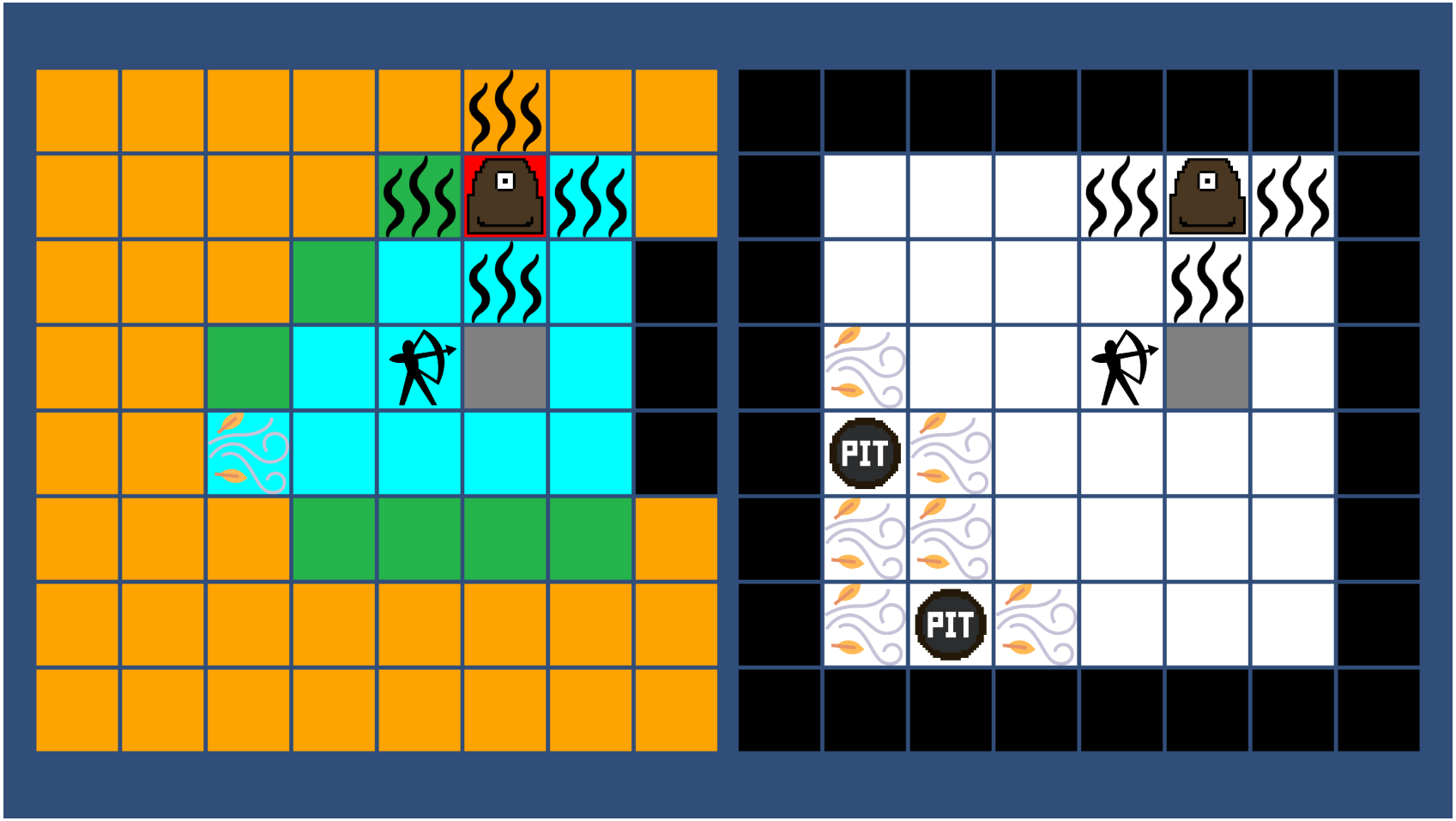


Figure 3: Turn 18.

Prototype: Wumpus World Multi-agent example

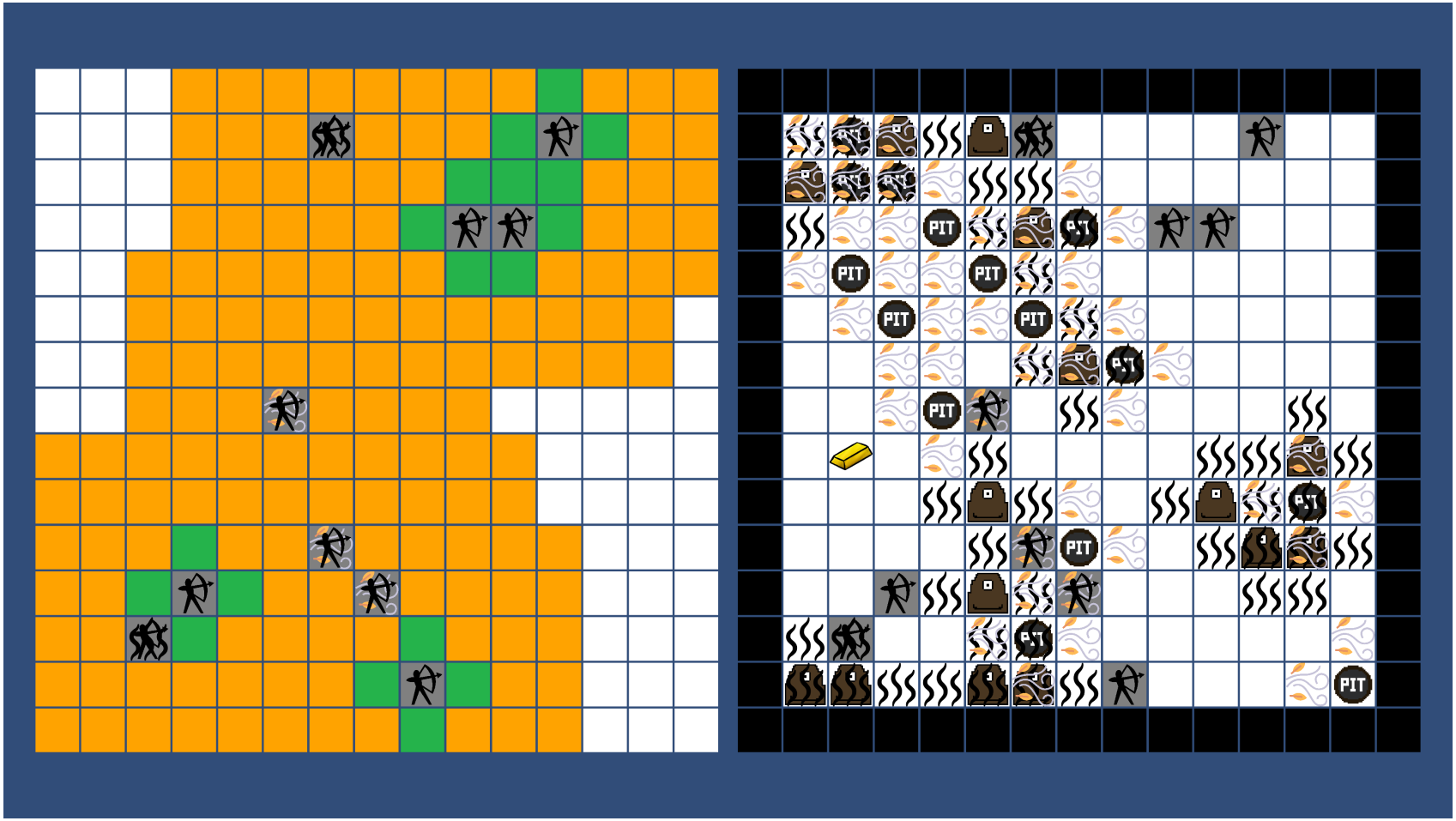


Figure 4: Turn 1.

Prototype: Wumpus World multi-agents example

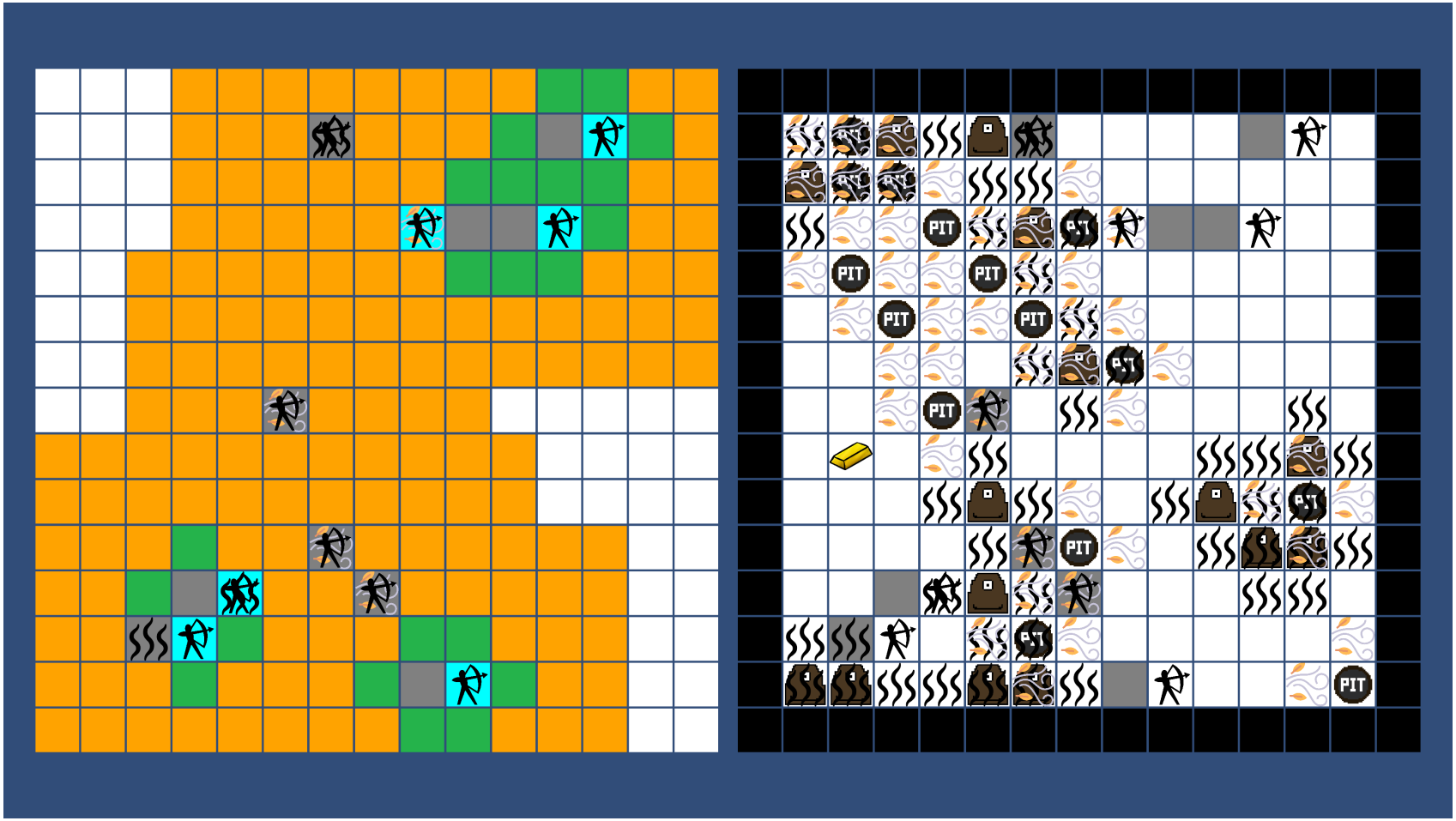


Figure 4: Turn 2.

Prototype: Wumpus World multi-agents example

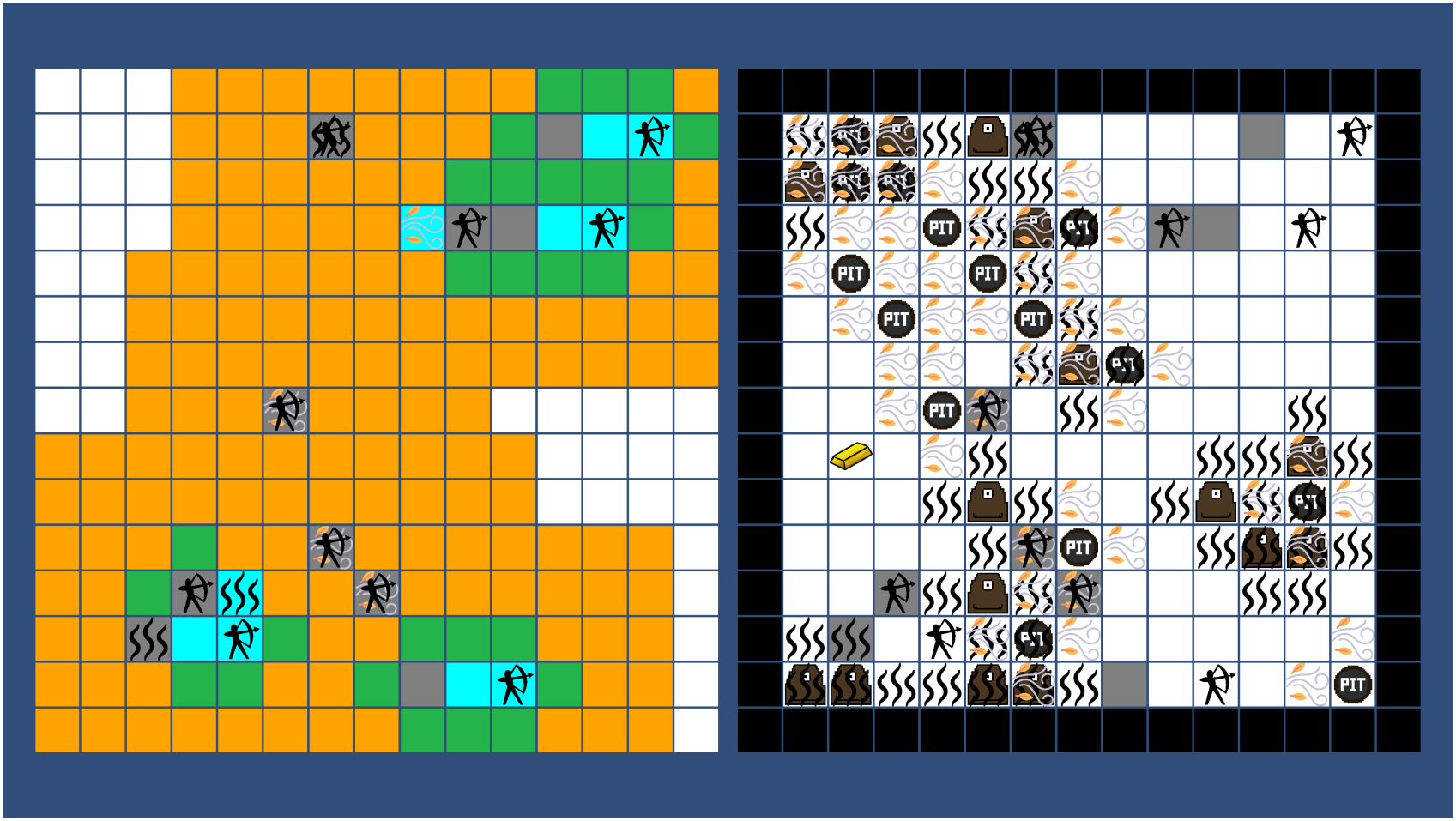


Figure 4: Turn 3.

Prototype choice

Why took the Wumpus World example?

- Prolog designed example
- Quickstart example
- It is "popular" and documented
- It is easily expandable

Is it perfectly suited for what we are trying to do?

Prototype choice

Why took the Wumpus World example?

- Prolog designed example
- Quickstart example
- It is "popular" and documented
- It is easily expandable

Is it perfectly suited for what we are trying to do? **Probably not...**

Prototype choice

Why took the Wumpus World example?

- Prolog designed example
- Quickstart example
- It is "popular" and documented
- It is easily expandable

Is it perfectly suited for what we are trying to do? **Probably not...**

What do we want to do next?

Current work

- Improve (or change) the prototype to enable even more complex behaviour, especially for multi-agents
- Better formalise the creation and interaction with ontologies

What do we want to do next?

Current work

- Improve (or change) the prototype to enable even more complex behaviour, especially for multi-agents
- Better formalise the creation and interaction with ontologies

Future work

- Integrate and test the AI in Wako Factory's commercial game
- Make logic programming easier to use for game designers outside academia

What do we want to do next?

Current work

- Improve (or change) the prototype to enable even more complex behaviour, especially for multi-agents
- Better formalise the creation and interaction with ontologies

Future work

- Integrate and test the AI in Wako Factory's commercial game
- Make logic programming easier to use for game designers outside academia

Thank You!

sylvain.lapeyrade@uca.fr

References I

Denise Angilica, Giovambattista Ianni, and Francesco Pacenza. Declarative ai design in unity using answer set programming. In *2022 IEEE Conference on Games (CoG)*, pages 417–424, 2022. doi: 10.1109/CoG51982.2022.9893603.

Andreas Brännström and Juan Carlos Nieves. UnityIIS: Interactive Intelligent Systems in Unity. 12 2021. URL `https://git.io/JMpzr`.

Francesco Calimeri, Stefano Germano, Giovambattista Ianni, Francesco Pacenza, Simona Perri, and Jessica Zangari. Integrating rule-based ai tools into mainstream game development. In *RuleML+RR*, 2018.

Richard Evans and Emily Short. Versu, 2014. URL `https://versu.com`.

References II

Andrea Gatti and Viviana Mascardi. Towards VEsNA, a framework for managing virtual environments via natural language agents. *Electronic Proceedings in Theoretical Computer Science*, 362: 65–80, jul 2022. doi: 10.4204/eptcs.362.8. URL `https://doi.org/10.4204%2Feptcs.362.8`.

Ian Horswill. Mkultra (demo). *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 11(1):223–225, 2015. URL `https://ojs.aaai.org/index.php/AIIDE/article/view/12776`.

Ian Millington. *AI for games*. Taylor & Francis, a CRC title, Boca Raton, third edition edition, 2019. ISBN 978-1-138-48397-2.

Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson series in artificial intelligence. Pearson, Hoboken, fourth edition edition, 2021. ISBN 978-0-13-461099-3.

References III

Andrey Simonov, Aleksandr S. Zagarskikh, and Victor Fedorov. Applying behavior characteristics to decision-making process to create believable game ai. *Procedia Computer Science*, 2019.

Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):619–649, July 1991. ISSN 0004-5411, 1557-735X. doi: 10.1145/116825.116838. URL `https://dl.acm.org/doi/10.1145/116825.116838`.

David S. Warren. Programming in tabled prolog, 1999.

Georgios N. Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. Springer International Publishing : Imprint: Springer, Cham, 1st ed. 2018 edition, 2018. ISBN 978-3-319-63519-4. doi: 10.1007/978-3-319-63519-4.