3.5 AI for Speedrunning

Michael Cook (King's College London, GB), Maren Awiszus (Viscom AG – Hannover, DE), Filippo Carnovalini (VU – Brussels, BE), M Charity (New York University, US), and Alexander Dockhorn (Leibniz Universität Hannover, DE)

Speedrunning refers to a collection of related activities where people play games under specific conditions – usually trying to complete a game as quickly as possible, but sometimes trying to complete it with certain restrictions (e.g. while blindfolded), variations (e.g. randomisers which alter the structure of an otherwise static game), or other feats (e.g. two players sharing a single controller). Speedrunning is a very popular subculture within games: the official portal *speedrun.com* reports 20m annual visits to their site, which hosts over 4.7m individual speedruns across 43.2k games [1].

Speedrunning remains vastly understudied within game AI research, despite the obvious parallels between game-playing AI research and time-optimised game-playing. Interestingly, one of the few pieces of academic writing about speedrunning in games comes from a Dagstuhl publication [2], with some studies existing from a sociological or cultural perspective outside of AI [3]. This working group set out to discuss the many problems that exist within speedrunning for AI researchers to tackle, and then to concretely implement a prototype platform for speedrunning research with AI systems.

We began by discussing the current state of speedrunning and identifying where there was potential for impact. The speedrunning community is inventive and resourceful, and already do a lot of work that would be considered research-grade in some fields: randomisers, for example, procedurally modify games to make them unpredictable to play, while retaining consistency in terms of pacing, flow and complexity. We also discussed *tool-assisted speedruns* (TAS), where speedruns are executed by a computer replaying pre-defined commands (not competing with human speedrunners). This allows speedrunners to perform tricks requiring superhuman skill, but each TAS must be made by hand.

Speedruns, no matter what form they take, usually exploit *glitches* in games to skip content or progress faster. These glitches take on many forms, including manipulating data in memory, forcing physics simulations into edge case scenarios, and causing simultaneous execution of code through multiple inputs. Many of the most popular AI environments for game-playing in the past decade are competitive, meaning they are ranked on winrate rather than time taken. For single-player games used as AI environments, such as DOOM, the reward signal for the discovery and use of glitches is likely too weak for most AI to use. For this reason, we chose to use the workgroup to build testing environments for single-player, open-source and speedrunnable games, so that we can investigate this problem space further in the future. In the next section we describe our chosen platform, the game engine *PICO-8* and the game *Celeste*.

3.5.1 Celeste and PICO-8

PICO-8 is a *fantasy console* – a type of game engine specifically designed to be highly constrained, often mimicking the hardware restrictions in consoles from the 1990s and 1980s. PICO-8 is perhaps the most popular example of this. Its restrictions include a 128x128 pixel screen, a palette of 16 colours, a maximum size of 32kb for games, and a limit of 256 game sprites (see Figure 8). All PICO-8 games are open-source, and are distributed online through a BBS-like system within PICO-8 itself.

Figure 8 The sprite design interface.

Celeste, by EXOK Games, is a "hardcore platformer" released in 2017. The primary mechanic in the game is dashing – the player can press a button to dash once in any direction, including in the air. This is reset when they are touching the ground again. Celeste became popular with speedrunners due to its difficulty level and the high skill ceiling of its controls. Celeste was expanded into a full game and released in 2018, where it won numerous awards and sold millions of copies. The full game is also beloved by speedrunners, and has many specific dedicated speedrun mods and extensions made for it.

We chose PICO-8 as a target domain due to its openness and the ease with which games can be instrumented and analysed. Celeste was an obvious target for us because its prominent position in the speedrunning community and its many known glitches and exploits. However, during the working group we discovered *Celeste Tech Training*, a PICO-8 game made specifically to teach speedrunners how to perform certain tricks. We modified this game to strip out unnecessary functionality, and used its focused levels to test our prototypes on. Figure 9 shows the first level of this game, which teaches a technique called *spike jumping*. Spike jumping allows the player to jump on a specific part of a spike floor without being hurt.

3.5.2 Approaches

We implemented three different systems for replicating the spike jump technique in *Celeste Tech Training* (CTT). Our first system was built into the code – it simulates virtual inputs and can load and save game states using custom code. This is the least flexible solution as it needs to be rewritten for different games, but it is the most portable – it is self-contained within the cart and does not require any external tools.

The second solution leverages Celia [4], a LUA software designed to facilitate the creation of TASs of PICO-8 games. By modifying the source code of the project, the software was adapted to automatically create TASs of a simplified Celeste level which requires a spike

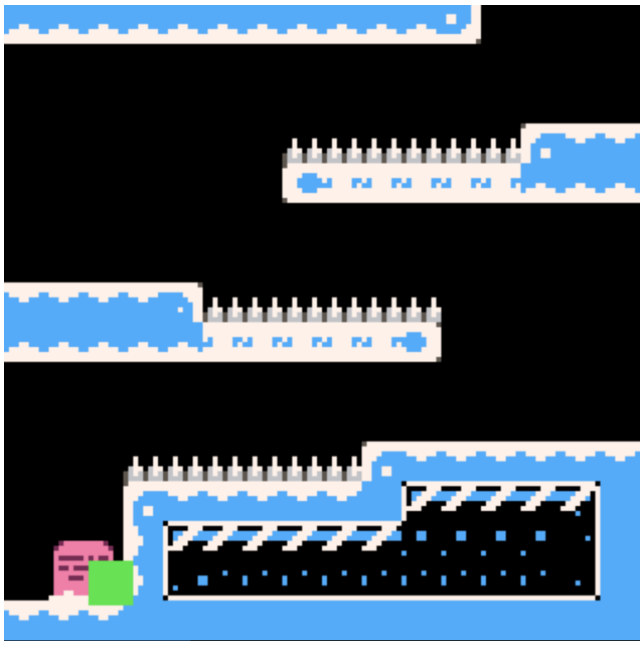

Figure 9 Modified version of CTT.

jump to be beaten (see Figure 9). We implemented a random agent that adds random inputs every fifth frame of TAS. The distance of the character from the goal position (beyond the spikes section that requires a spike jump to be cleared) served as an objective function. Whenever a new shortest distance was achieves, the TAS was saved, providing record of how it is possible to reach that distance. In our tests[8], the random agent was unable to reach the destination point, but it managed to perform the initial part of the spike jump: it jumped on the correct pixel at the corner of the spikes, but failed to then perform a dash at the correct moment to clear the needed distance. A Reinforcement Learning based agent could probably fare better than this naïve random agent, providing more adaptability over our first approach.

For a more general approach, we designed a Python interface to work with the Celia software. With the *pynput* library, this approach used keypresses and Celia command shortcuts (i.e. loading the TAS files, skipping frames) to play the PICO-8 games frame-by-frame. Evaluation for this approach would involve retrieving screenshots from the game through the Celia software. With the frame manipulation, the game could also be reset to earlier states for tree-searches of optimal paths and keystrokes. With this approach, an AI-generated speedrun could be made for any PICO-8 game that could be loaded into Celia; without manipulating the source code of the game or Celia itself.

We tested this methodology on three different games retrieved from the PICO-8 community BBS[9]: Get Out of this Dungeon, treeboi_test, and Witch Loves Bullets. With all three games, randomly made TAS files were successfully generated, loaded, and played in the Celia software. Future work would look to using tree-search methods such as A* to create speedruns.

[8] The modified Celia code is available at `https://github.com/Facoch/Celia`
[9] `https://www.lexaloffle.com/bbs/?cat=7`

3.5.3 Further Work

Our next steps are to clean up and open source these systems, along with publishing our initial survey of the speedrunning landscape with respect to AI. Beyond that, we believe Celeste in PICO-8 represents a good domain for an AI competition. Designing the framework and rules for this competition will also help us clarify what challenges are most interesting, and begin to grow academic interest around this area.

References

1 Speedrun.com – About. http://www.speedrun.com/about Accessed 3 July 2024.
2 Manuel Lafond. *The complexity of speedrunning video games*. In 9th International Conference on Fun with Algorithms (FUN 2018). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.
3 Scully-Blaker, Rainforest. *Re-curating the Accident: Speedrunning as Community and Practice*. Masters thesis, Concordia University, 2016.
4 gonengazit/Celia. https://github.com/gonengazit/Celia Accessed 9 July 2024.

3.6 Skill-Discovery in (Strategy) Games

Alexander Dockhorn (Leibniz Universität Hannover, DE), Manuel Eberhardinger (Hochschule der Medien – Stuttgart, DE), Chengpeng Hu (Southern Univ. of Science and Technology – Shenzen, CN), and Matthias Müller-Brockhausen (Leiden University, NL)

Strategy games present a unique challenge in artificial intelligence (AI) research. They can broadly be classified into two types: turn-based and real-time strategy games. Both types typically require the player or AI to manage multiple units or resources, often with incomplete information about the opponent's actions. The large branching factor and long game duration make it difficult for AI to explore all possible strategies, which is further complicated by the need to plan several moves ahead. The state-of-the-art methods in AI for strategy games include search-based algorithms and reinforcement learning (RL), but these often rely on human-defined strategies or subgoals, limiting their scalability and generalizability.

While the work on AlphaStar [2, 3] and OpenAI Five [5] have shown that it is possible to train strong AI agents for complex games such as Starcraft 2 and Dota 2, both works required massive amounts of compute resources until satisfying results have been achieved. For the purpose of speeding up the learning process, the working group on skill discovery in (strategy) games has been formed to evaluate the applicability of skill discovery methods to this special domain. We particularly emphasize works on skill discovery as part of RL algorithms. In this context, skill discovery refers to identifying and learning sub-policies or strategies that can be applied to achieve or identify specific subgoals within a game, thereby enabling more efficient and scalable AI systems.

3.6.1 Preliminaries of Skill Discovery

Skill discovery in AI remains an open problem, particularly when it comes to discovering skills without human intervention. The interpretation of what constitutes a "skill" in a given context is still unclear, making it challenging to develop a unified approach to skill discovery.