Analysis and Generation of Flow in 3D Jump'n'Run Games

Tobias Brandner
Julius Maximilian University
Würzburg, Germany
tobias.brandner@gmx.de

Marc Mußmann
Julius Maximilian University
Würzburg, Germany
ma_mussmann@posteo.eu

Sebastian von Mammen
Julius Maximilian University
Würzburg, Germany
sebastian.von.mammen
@uni-wuerzburg.de

***Abstract*—Playing Luigi in the original Super Mario Bros feels completely different to playing Mario. Why is that so? We propose an analytical way to design movement in 3D Jump'n'Runs. To this end, we analyzed the movement parameters of three influential games in the genre–Super Mario 64, Banjo Kazooie and Donkey Kong 64–and visualized their evolution with graphs. To compare the findings and further explore the parameter space, we created our own 3D Jump'n'Run game in Unreal Engine 5 supporting a broad set of movement mechanics. Constrained by the concrete movement parameters, we procedurally generate levels by means of Blender's Geometry Nodes and evaluate the resulting flows of movement in a preliminary study. We believe this approach achieves a high control and explainability for movement mechanics and level design.**

***Index Terms*—game feel, game analysis, platform games, 3D, procedural content generation**

I. Introduction

The Super Mario Bros series is one of the best-known and most popular platformers on the market. But why are they so much fun to play? In his book *Game Feel*, Steve Swink describes this phenomenon as game flow with the words:

> Real-time control of virtual objects in a simulated space, emphasizing interactions through polish [1].

Game flow is therefore based on three key elements: Real-time control, spatial simulation, and polish. With respect to 3D Jump'n'Runs, this means: Reasonable control and response of the player character, a game world that matches the player's movement parameters, e.g., his jump height and distance, and audio-visual cues that emphasize the player character's interactions in the world. To investigate and analyze movement flow, we first developed a broad set of movement mechanics typically found in 3D Jump'n'Runs. To choose the right parameters for these mechanics, we analyze three influential games in the genre: Super Mario 64 [2], Banjo Kazooie [3], and Donkey Kong 64 [4]. Based on this analysis we added our own configuration called Boss'n'Run. To evaluate these four configurations, we procedural generated a level based on each individual set of movement parameters. The generated level geometry fits the movement parameters and ensures the playability of the respective configuration. This enables fast evaluation of different sets of movement parameter variations.

II. Related Work

Designing the optimal movement mechanics for a game is an elaborate, iterative task, which can be supported by means of an analytical underpinning: Movement mechanics are built upon parameters. For example, a jump is defined by gravity, velocity, height and distance. All of these parameters have been measured, analyzed, and compared with each other in a 2D context [5], [6]. We propose a framework to investigate 3D Jump'n'Runs.

Movement parameters are tightly coupled to the surrounding game world and are represented in their platform and obstacle layout [7]. This makes it hard to explore vastly different values as it involves changing the level layout. In contrast to manual level creation, procedural generation can be used to create levels based on a set of parameters. These can determine the level layout and difficulty of platformer games [8]. Additionally, continuous Jump'n'Run level generation is possible by sewing together pre–generated, mechanic–focused scenes [9]. Concerning 3D level generation, there exist several solutions such as the commercial 3D animation software Houdini [10]. But also the open source 3D software Blender recently introduced a feature called Geometry Nodes, which provides tools that enable a node-based procedural modeling workflow [11]. This workflow allows for fast procedural parametric concept design of complex 3D geometry [12].

III. Methodology

We designed and implemented a 3D Jump'n'Run[1] with a broad set of movement mechanics in *Unreal Engine 5* as well as a procedural level generation system created with Blender's Geometry Nodes. The level generation system is embedded into the Unreal engine using the Altermesh plugin [13].

A. The player character

A 3D avatar was modeled in Blender, rigged with *AccuRig*[2] and animated using Unreal's re-targeting system. Next to the usual movement mechanics, we implemented three different jump mechanics as well as a climbing mechanic. An overview of all the implemented movement states and their transitions can be seen in figure 1.

[1]https://brandnerkasper.itch.io/bossn-run
[2]https://actorcore.reallusion.com/auto-rig

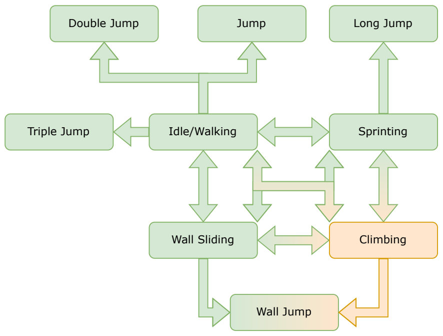


Fig. 1: Hierarchically structured taxonomy of all movement states that the player character can enter in the game world.

Similar to Super Mario 64, we implemented a triple jump and a long jump: After performing a regular jump and landing the player, there is a short time window to jump again to perform a double, and then a triple jump resulting in jumps with greater height. Jumping while running results in a long jump, which covers more distance but less height. If the player jumps against a wall, the character enters a sliding state with reduced gravity. Instigating a jump in this state yields a wall jump. The wall jump catapults the character upwards but also away from the wall, whereby the launch angle can be controlled. On certain walls the character is allowed to climb. While climbing, jumping also results in a wall jump. All states are polished through animations and blendspaces. Furthermore, we added particle effects to plausibly anchor the character's movements in the world. For example, while the character is running small dust clouds are spawned beneath his feet. We implemented a rather common dual-stick joypad control, where the left stick moves the character and the right one the camera.

B. The level

There exist multiple approaches to create procedural levels or structures. We use a hierarchical approach for generating a coherent level, that is guided by a spline curve. This hierarchy consists of higher-level, connected structures formed from sequentially connected sections. The implemented sections are tailored to the four different movement mechanics of climbing, wall jumps, long jumps, and triple jumps, as their interplay yields the greatest challenges. For the long jump section, we placed two platforms far apart. The closer to the long jump distance, the more difficult is the leap. For the triple jump section, three platforms are placed between a starting platform and a target platform. Here, the difficulty scales with the distances of the three in-between platforms and their difference in elevation. Both sections can be seen in figure 2. The climbing and wall jump sections are shown in figure 3. The climbable geometry is generated around larger, pillar-like structures. Wall jumps can be performed on a number of walls placed between two platforms similarly to the long jump section. The level generation is constrained by the movement parameter presets, ensuring that any level is solvable.

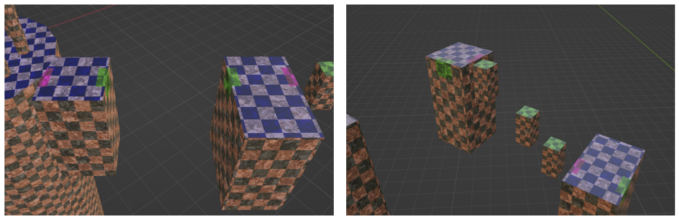


Fig. 2: Left: Section related to the long jump mechanic. Right: Section related to the triple jump mechanic.

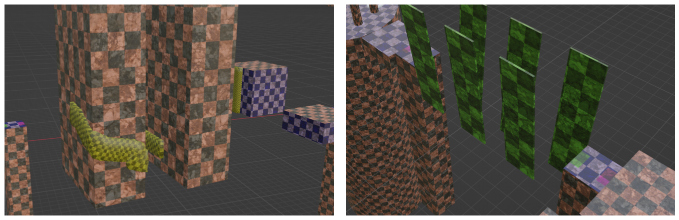


Fig. 3: Left: Section related to the far climbing. Right: Section related to the wall jump mechanic.

IV. Analysis

We collected movement data from the aforementioned games, compared and plotted them. For example, investigating a simple walking movement of a character, we look at *a)* its acceleration *b)* maximum speed, and *c)* deceleration.

A. Gathering data from other games

Since we focus on 3D Jump'n'Runs, we chose three influential games of the genre: Super Mario 64 (SM64), Banjo Kazooie (BK), and Donkey Kong 64 (DK64). We measured the movement parameters such as maximum running speed or jump height using recorded video footage from the respective game [5]. To time the values as exactly as possible, we analyzed the recorded footage frame by frame using Shotcut[3]. For the maximum running speed, we captured footage of the character running at full speed for a given estimated distance and time. We then calculated acceleration/deceleration by analyzing video footage of the character starting and stopping. In SM64, for example, the acceleration phase is indicated by a particle system, some dust clouds, that stop spawning once the character has fully accelerated. Knowing the maximum running speed and the time it takes the character to reach it, we can calculate acceleration/deceleration using eqn. 1.

$$a = \frac{\Delta v}{\Delta t} \tag{1}$$

For the jump height and gravity, we analyzed the character jumping/falling over a certain estimated distance and tracking his time in the air. Using the formula for projectile motion and rearranging it, we can calculate gravity with eqn. 2.

$$g = -\frac{2h}{t^2} \tag{2}$$

[3]https://shotcut.org/

Knowing the gravity and the estimated jump height, we can calculate the jump velocity of the character using eqn. 3 [14].

$$v_j = \frac{h - \frac{1}{2}gt^2}{t} \tag{3}$$

Related work normalized movement parameters by the characters' height and width to better compare them across games. We use metric units to ease development and get a better understanding of the values. Therefore, the data collected is influenced by estimates to some degree, since none of these games have a reference object in metric units.

B. Comparing movement parameters

After collecting and calculating the movement parameters of the four games, we visualize the mechanics in graphs and compare the parameters in tables. We consider the walk movement first, as shown in in figure 4.

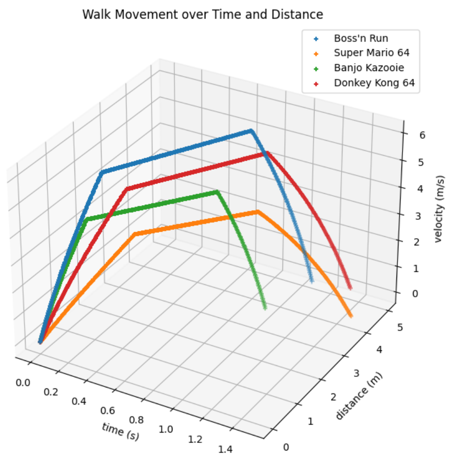


Fig. 4: The graph shows the distance traveled over time and the resulting velocity. The walking analysis considers three phases: Accelerating from a previous halt, moving at full speed for $0.5s$, and decelerating to a halt.

For our own preset, Boss'n'Run (BnR), we combined the highest acceleration, speed, and deceleration of the other three games. SM64 has the lowest speed, acceleration and deceleration. It also takes the longest amount of time to reach maximum speed, and to come to a halt. DK64 takes the longest distance traveled to reach the maximum speed and to come to a halt. BK takes the least time and the least distance to reach the maximum speed, and to come to a halt. The parameters are summarized and compared in table I.

Looking at figure 5, we see how the jump differs between the four games. BnR has the highest speed on the ground and therefore can jumps the farthest. SM64 features the highest gravity and the lowest ground speed. This results in the shortest jump distance. Yet, it still achieves the highest jump by overcoming (the greatest of all) gravity for only a short period of time. DK64 has the lowest jump height and shortest airtime. BK has the lowest gravity and jump speed and

TABLE I: Movement parameters for walking. For speed, acceleration, and deceleration, highest values are marked in red and lowest values in blue. For time and distance to reach maximum speed, lowest values are marked in red and highest values in blue.

	BnR	**SM64**	**BK**	**DK64**
Velocity ($\frac{m}{s}$)	6	4	4.5	5.4
Acceleration ($\frac{m}{s^2}$)	20	8	18	12.8
Deceleration ($\frac{m}{s^2}$)	20	8	18	12.8
Time (s)	1.1	1.5	1.0	1.35
Distance (m)	4.8	4	3.38	4.98

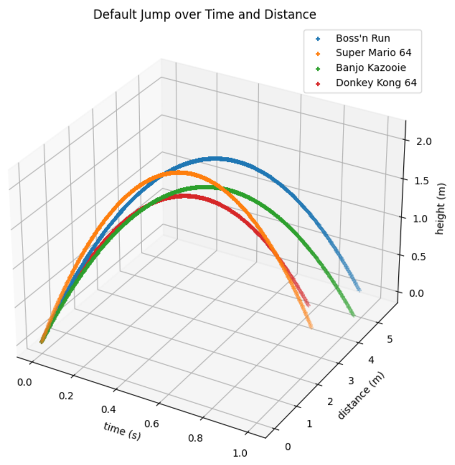


Fig. 5: The jump height and distance over time of a character leaping at full speed.

remains the longest time in the air. The jumping paramters are summarized and compared in table II.

TABLE II: Default jump parameters of the four games BNR, Super Mario 64, Donkey Kong 64, and Banjo Kazooie. For all parameters, the highest values are marked in red and the lowest in blue.

	BnR	**SM64**	**BK**	**DK64**
Gravity ($\frac{m}{s^2}$)	-18.6	-20.7	-14.0	-20.2
Ground Velocity ($\frac{m}{s}$)	6	4	4.5	5.4
Jump Velocity ($\frac{m}{s}$)	8.7	9.3	7.3	8.2
Height (m)	2.03	2.09	1.84	1.66
Time (s)	0.93	0.9	1.0	0.81
Distance (m)	5.6	3.6	4.5	4.4

V. Evaluation

We conducted a preliminary study with 14 participants–mainly Computer Science students. They were tasked to play and evaluate the four different movement presets and according levels. The resulting difference in level generation is shown in figure 6. The participants were asked three questions:

1) How do you like the jump mechanics? (Rated from 1 (bad) to 5 (good))
2) How is the overall Game Flow for you? (Rated from 1 (poor) to 5 (fluid))
3) How difficult was the level for you? (Rated from 1 (easy) to 5 (difficult))

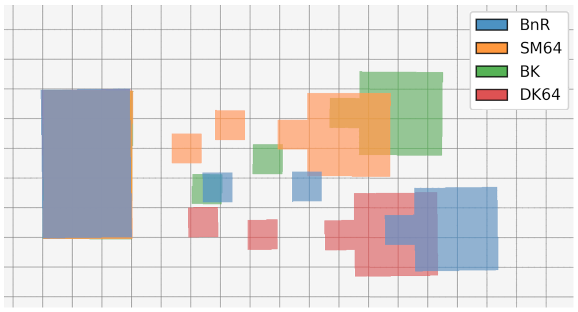


Fig. 6: Top view of the triple jump section, generated based on movement presets: Boss'n'Run (BnR), Super Mario 64 (SM64), Banjo Kazooie (BK) and Donkey Kong 64 (DK64)

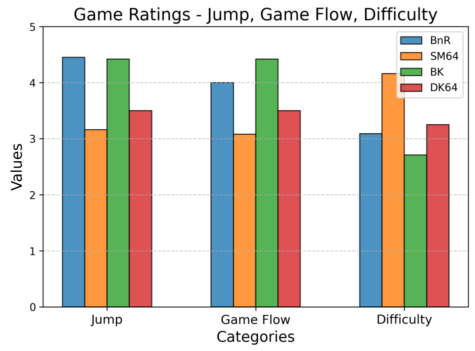
We have summarized the results in figure 7. One can see relationships between the three evaluation categories. For instance, the ratings of the jump mechanics and the game flow coincide well across all four games. Similarly, the difficulty ratings are inversely proportional to the jump and flow ratings for the three games BnR, SM64, and BK. In particular, BK received the highest jump/flow ratings and the lowest difficulty rating. BnR received similar ratings, albeit a slightly lower flow value. SM64 is opposing these two games, achieving the lowest jump and flow ratings (but coinciding well) and reporting the greatest experienced difficulty. Interestingly, all of DK64's ratings assume the middle ground of the four examples, and the effect of opposing jump/flow vs. difficulty is still there but hardly noticeable. DK64's ratings, thus, seem close to the demarcation where the ratio of positive jump/flow and opposing difficulty ratings flips.

Although these results suggest a consistent relationship between jump/flow and difficulty, the study is very preliminary and has several shortcomings. The correlation between jump and flow ratings could stem from a strong dependency of the two measured variables and not imply a causal relationship. The number of participants was rather low and not all participants were exposed to each condition, which led to even fewer absolute ratings. Questionnaires with more participants and more refined questions might resolve this issue. We only focused on traditional 3D Jump'n'Run's with a more similiar movement parameter space. Exploring a wider range of games where the calculation of mathematical correlation values yielding greater certainties should, therefore, be pursued. The bigger result space in turn could then be used to find optimal parameters, via a grid search.

VI. Conclusion & Future Work

We measured and compared the movement mechanics of three popular 3D Jump'n'Run games. We presented a hierarchical approach to the design of 3D Jump'n'Run levels, which adhere to according movement parameters. In a preliminary study, we asked 14 participants to rate the jump mechanics, the flow, and the difficulty of instances of the three measured move sets and an additional one, which implemented the most vivid behavior. This early study suggests that flow and positively

Fig. 7: The results of the preliminary study for the four movement presets: Boss'n'Run (BnR), Super Mario 64 (SM64), Donkey Kong 64 (DK64), and Banjo Kazooie (BK). Evaluating the feel for the jumping mechanics, overall flow of the game and the perceived difficulty of the level.



evaluated jump mechanics go hand in hand and that there is an inverse relationship between flow and difficulty. While more research is required to corroborate these findings, we believe that the overall approach to combine an analytical design of move sets, to complement them with procedurally generated level designs, and to select and refine parameters based on user tests provides a sound testbed for further investigations.

References

[1] S. Swink, *Game feel: a game designer's guide to virtual sensation*. CRC press, 2008.
[2] Nintendo, "Super Mario 64," https://www.nintendo.de/Spiele/Nintendo-64/Super-Mario-64-269745.html, accessed: 2024-03-24.
[3] Rare, "Banjo Kazooie," https://www.xbox.com/de-DE/games/store/banjo-kazooie/bsjg7ttswvj2, 1998, accessed: 2024-03-24.
[4] ——, "Donkey Kong 64," https://www.nintendo.de/Spiele/Nintendo-64/Donkey-Kong-64-269459.html, accessed: 2024-03-24.
[5] M. Fasterholdt, M. Pichlmair, and C. Holmgård, "You say jump, i say how high? operationalising the game feel of jumping." in *DiGRA/FDG*, 2016.
[6] Game Maker's Toolkit, "Why does celeste feel so good to play?" https://www.patreon.com/posts/28582857, accessed: 2024-03-24.
[7] G. Smith, M. Cha, and J. Whitehead, "A framework for analysis of 2d platformer levels," in *Proceedings of the 2008 ACM SIGGRAPH symposium on Video games*, 2008, pp. 75–80.
[8] R. R. Wehbe, E. D. Mekler, M. Schaekermann, E. Lank, and L. E. Nacke, "Testing incremental difficulty design in platformer games," in *Proceedings of the 2017 CHI conference on human factors in computing systems*, 2017, pp. 5109–5113.
[9] M. C. Green, L. Mugrai, A. Khalifa, and J. Togelius, "Mario level generation from mechanics using scene stitching," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 49–56.
[10] Side Effects Software Inc., "Houdini," https://www.sidefx.com/products/houdini/, accessed: 2024-04-04.
[11] J. van Gumster and J. Lampel, "Procedural modeling with blender's geometry nodes: A workshop on taking advantage of the geometry nodes feature in blender for procedural modeling," in *ACM SIGGRAPH 2022 Labs*, 2022, pp. 1–2.
[12] M. Denk, J. Mayer, H. Völkl, S. Wartzack *et al.*, "Procedural concept design with computer graphic applications for light-weight structures using blender with subdivision surfaces," in *DS 119: Proceedings of the 33rd Symposium Design for X (DFX2022)*, 2022, pp. 1–10.
[13] Aechmea Studios, "Altermesh," https://altermesh.com/, accessed: 2024-04-04.
[14] GDC, "Math for game programmers: Building a better jump," https://www.youtube.com/watch?v=hG9SzQxaCm8&t=461s, accessed: 2024-03-24.