Agent-based crowd simulation for building plan

Kaleab Belete
University of California, Berkeley
Berkeley, U.S.
xinwei_zhuang@berkeley.edu

Xinwei Zhuang
University of California, Berkeley
Berkeley, U.S.
xinwei_zhuang@berkeley.edu

Tristan Streichenberger
University of California, Berkeley
Berkeley, U.S.
tristanstreich@berkeley.edu

Gregoria Millensifer
University of California, Berkeley
Berkeley, U.S.
gregoriiaaa@berkeley.edu

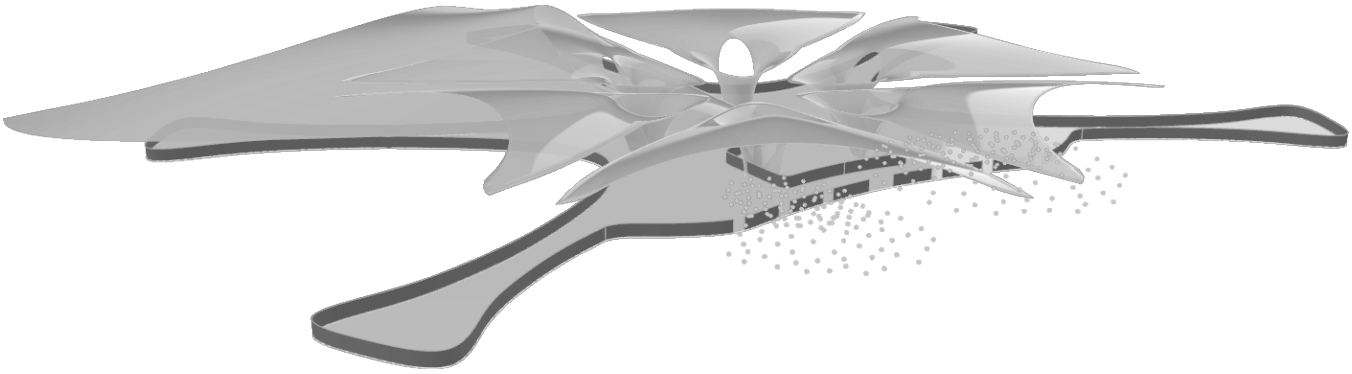

Figure 1: Crowd dynamics in Daxing Airport.

ABSTRACT

Circulation simulation is critical for both architecture design and urban planning, which has a great impact on the efficiency of the common daily life as well as emergencies such as fire incidents and terrorism. We propose an agent-based circulation simulation with personalized characters to visualize, evaluate, analyse and optimize the building plan. We use ray tracing to detect the furthest direction one agent can reach, and use Russian Roulette for probability of turning into a specific direction. We introduce swarm algorithm to the crowd, but for each agent we introduce randomness by parameterizing different behaviours. Then for each building plan, we provide goals, for the crowd to reach a specified destination, and use the converge time as evaluation of the effectiveness of the building plan. Our hypothesis is that the building plans with curvature walls will have a better performance regarding the circulation efficiency.

KEYWORDS

Crowd simulation, Crowd dynamics, Agent based modelling, Collision Avoidance, Rock dynamics

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Kaleab Belete, Xinwei Zhuang, Tristan Streichenberger, and Gregoria Millensifer. 2022. Agent-based crowd simulation for building plan. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

It is crucial when architects make the design decisions for choosing the best circulation plan, especially for large activities. Failure in circulation design can cause problems ranging from making it confusing to find ways and locating to Stampede, and can be critical when unexpected incidents happens [7]. This study aims to provide a evaluation tool for circulation in complex building settings by stimulating behaviors of crowds. We use Daxing airport as a case study, for it serving as a model for the efficient circulation for airport. We perform our simulation within the airport, by initializing groups of people at the entrance, and set different destination within the population.

The group will search for the direction with maximum visual depth, proceeding to the goal position with behavioral rules within a group. We use particle simulation with swarm algorithm to generate the crow behaviour, we then incorporate randomness by setting different behavior within the group, including the preference of turning at a crossroad, the distance to the majority of people, etc., to perform circulation simulation within a building, and use the simulation to evaluate the design of the building plan. We aim to simulate how people behave in crowds with different environments,

and more to accurately model and simulate crowd behavior and inform design and planning decision.

2 RELATED WORK

Many studies investigated how to accurately model the crowd behaviour with real time simulations [5][6][8][10]. Popular approaches includes flow-based approach [3], entity-based approach, and agent-based approach [11].

Recently, Li et al. [4] (2022) investigated the influence of geometric layout of exit on escape mechanism of crowd.

3 METHODOLOGY

We integrated several algorithms to perform the crowd simulation. In the agent level, we used ray-tracing based collision avoidance to find the direction with maximum depth. We use Russian Roulette to pick the actual turning direction based on the depth information. In the crowd simulation, We perform used flocking algorithm to perform crowd dynamics, which is a combination among cohesion, separation and alignment. We also introduce randomness in the group with different behaviour, such as aggressive, shyness, to perform a comprehensive group behaviour. Finally, we used path finding to find the nearly-optimal path for the agents to follow.

3.1 Collision Avoidance

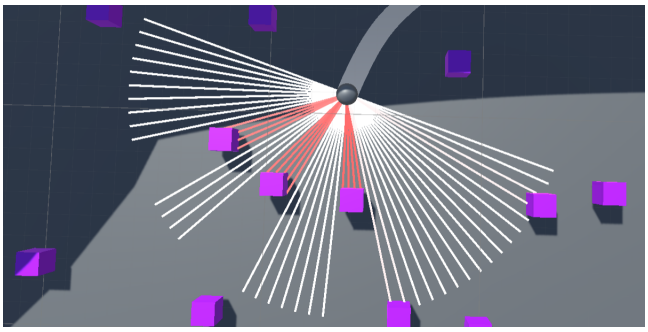

Figure 2: Rays detecting objects in a person's path

$$s \sum_{r} \left(\theta_r - \frac{\pi}{2}\right)\left(1 - \frac{d_r}{D}\right)^p$$

We added collision avoidance as an individual behavior for each of the people by implementing ray casting. On each update, a person sends out rays out in a fan centered on their current velocity. For each ray, a nudge angle is calculated based on the the above equation (to the right of the summation). θ_r is the ray's angle away from the current velocity. d_r is the distance to the collision. D is the max distance the ray checks. p is some exponential term. This all equates to an angle that is 90° away from the collision and is scaled down based on how far away the collision is. Close collisions will have a higher nudge angle and farther collisions will have smaller nudge angles. The exponential term amplifies this effect. The nudge angle is then summed for all the rays to get what angle the person should turn to to avoid the incoming obstacles. Finally, the sum is scaled down by s which is the turning speed. This makes the person not immediately jump to the angle they need to get to. Instead it takes several updates to get there, which makes their turning look smoother.

3.2 Crowd dynamics

We integrate individual agents using a rule-based flocking algorithm, separation, alignment and cohesion to simulate the crowd behaviour.

$$cohesion = \frac{1}{k}\sum_{k} rigidbody.transform.position - transform.position$$

$$separation = \sum_{i \neq j} rigidbody[i].transform.position - transform.position$$

$$alignment = \frac{1}{k}\sum_{k} rigidbody.transform.velocity - transform.velocity$$

$$steer = target.transform.position - transform.position$$

Separation is to maintain a certain distance for a object with its surrounding neighbours. If the distance between the neighbours is too close, the object will go to the opposite direction. We check the nearest k neighbours and find the combined direction. For cohesion the object will move towards the centroid of the group, again we are constraining the group to k nearest neighbours to accelerate the algorithm speed. For alignment the object is trying to adjust its speed with the average speed, and for steer the object will follow the nearest path towards its destination.

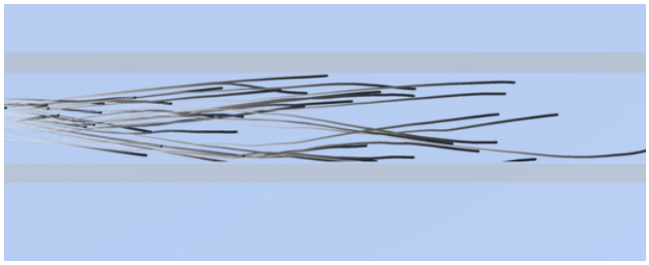

Figure 3: Flocking algorithm implementation with cohesion, separation and steer

3.3 Agent behaviour

The crowd are spawned at a specific segment (a stand in for doors at peak traffic) and begin moving in 2D space according to a preset rules: Movement is based on current position with a random movement factor included; Individual agent aims to maintain course for a noticeable subset of time steps to complete a goal; Individual agent has a innate repulsion factor to avoid collision but it is affected by randomness(the crowd tend to stay apart but not at the same distances), but also finds a fast path to their destination. For each agent, we modify the parameters according to their personality group by the following matrix[9].

$$\begin{bmatrix} V_{Shy} \\ V_{Normal} \\ V_{Aggressive} \end{bmatrix} = \begin{bmatrix} 0.03 & 0.06 & 0.02 \\ 0.22 & 0.04 & 0.32 \\ 0.11 & 0.04 & 0.13 \\ 0.28 & 0.16 & 0.41 \\ 1.05 & 1.07 & 1.02 \end{bmatrix}^T \begin{bmatrix} \frac{1}{13.5}(\text{Neighbour Dist - 15}) \\ \frac{1}{49.5}(\text{Max. neighbours - 10}) \\ \frac{1}{14.5}(\text{Horizon - 10}) \\ \frac{1}{0.85}(\text{Radius - 8}) \\ \frac{1}{0.5}(\text{Preferred speed - 1.4}) \end{bmatrix}$$

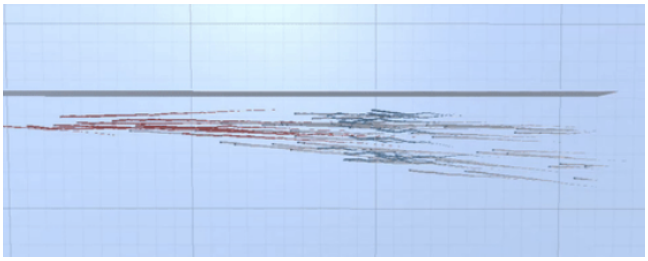

Figure 4: Different agent behaviours. Reds for aggressive personalities, greys for the average population, and blues are for the shy people.

3.4 Path planning

Planning requires us to take into account our goal as well as obstacles in our way and different planning decisions have different trade-offs between them. The core planning logic ties together everything else and produces the final action model. First we split the grid into discrete spaces and determinate which spots are walk-able and which are not. More granular splits will allow agents to move between tight spaces but this is memory intensive and may not always be needed. There are also more sophisticated algorithms we can use to create our grid mesh like a quad tree approach which on average can reduce the number of spaces we manage from n x n to n*log(n). This approach attempts to group large empty spaces into one large square rather than splitting the map evenly. For our situation the overhead of managing this adds up, as we have a considerable number of dynamic agents that are constantly moving, making it more reasonable to just have a constant grid we update in a single pass based on the agents positions. After we create our grid

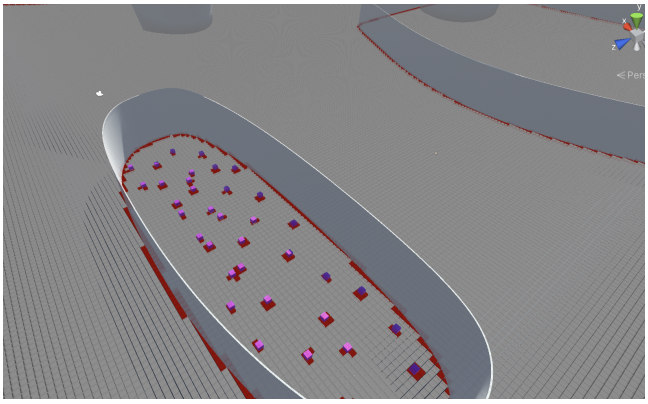

Figure 5: Grid mesh with walkable areas in gray

we find a path on the grid based on a path planning approach. For this assignment we explored a few approaches, the ones that stood out were Artificial Vector Fields/Potential Fields, A* path finding, and RRT/RRT* path finding. For Artificial Vector Fields/Potential Fields we split the map and all the objects in the map have a set pull or push force based on if they are an obstacle or goal (proportional to their size and distance) and we set the final force as the sum of all forces. This approach is good when working with a lot of agents, as you don't have to constantly check a large number of tiles for each agent, but it struggles with local minima so complex geometry has unpredictable results. A* path finding is a straight forward

Attraction: $$P_g = C\sqrt{|x - x_{\text{goal}}|^2 + |y - y_{\text{goal}}|^2}$$

Boundary Repulsion: $$P_{\text{HA}} = \frac{1}{\delta + \sum_{i=1}^{s}(g_i + |g_i|)}$$

Object Repulsion: $$p_{ij} = \frac{p_{\max}}{1+g}$$

Figure 6: Formulas

outward search where we use the Manhattan distance as the cost function. Moves are calculated based on the number of steps you have to take from the start point and the distance to the end goal. This algorithm is guaranteed to find a path if one exists. RRT/RRT* path finding is similar to A* but now we search randomly (with a maximum step distance out-words) and re-evaluate after each step. RRT* is a version of RRT that aims to provide more optimal paths by smoothing out the path through extended exploration and dynamic path linking. RRT* is more costly and for our use case unnecessary as path optimally is not important (people don't calculate optimal paths when walking). RRT performance was not too far from A* but it was less consistent (likely due to the random nature of the exploration). RRT benefits from high dimensionality but because we were moving across a 2d grid we did not get to leverage the boost in performance. Both RRT and RRT* are probabilistically guaranteed to find and a path if one exist. Ultimately a guided A* give the best performance for our use case. We put buffers around objects and dynamically updated the grid and queried new paths on the fly with collision checking. The paths are a bit rough but smoothness is not as important here as some other use cases, the performance improvement from foregoing smoothing was much more impactful. Lastly, after we set up our grid and structured our algorithm we had to decide how to update. The paths are updated on a timer but we also check for collisions, the agents are spawned with random properties like speed and vision length meaning we have to dynamically adjust the paths as they approach obstacles we did not plan for. For example if a person is at the goal already we stop early as to not crowd a given area.

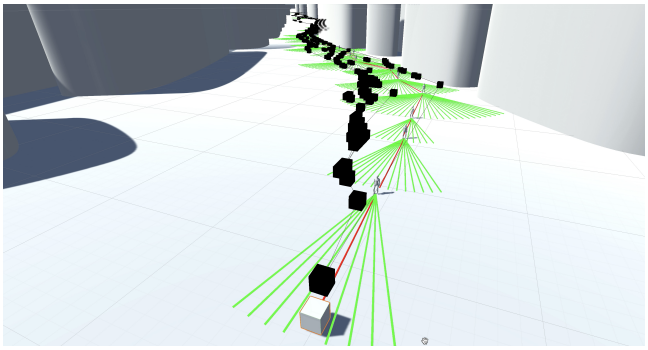

Figure 7: Dynamic Goal Stopping

3.5 Character Modeling and Animation

In order to add realism to our simulation, we also wanted to use character modeling for the people in the crowd. To do so, we used Blender, a free and open-source 3D computer graphics software toolset, to create 3D human models and animate their walk. As our goal is to simulate crowd behavior, we kept in mind the limitations of rendering multiple copies of a model simultaneously in our scene. So, we started off with a simple humanoid, then downloaded a complex character from Blender Studio, and finally found a simple humanoid that was not too complex but also still looked better than the basic humanoid we first created.

Using a very basic humanoid allows us to run the simulation faster and smoother because its lack of complexity means there are less polygons to render in the scene. The downfall is, of course, that it is very noticeably odd to the human eye.

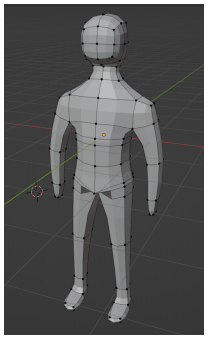

Figure 8: Very basic humanoid

In an attempt to make the simulation look more artistic, we downloaded Rain [2], a generic character rig offered for free from Blender Studio, and animated her walk in Blender. When we imported this character into our project, everything ran a lot slower and even making simple changes to the character's object properties would take a few minutes to update for all the instances. Thus, we decided to not use Rain for our project because, although it looks nice, the detail was unnecessary because we are more concerned in the crowd as a whole rather than a single person.

Finally, we found a compromise between artistic and efficient by downloading a simple humanoid on an open source website [1] and animating that. Because our end goal is to simulate crowd behavior, which requires a lot of character copies to be simultaneously present on the scene, it is more effective to use the simple humanoid rather than the complex character because the detail from the complex character was unnecessarily slowing down everything.

Figure 9: Complex character, Rain [2]

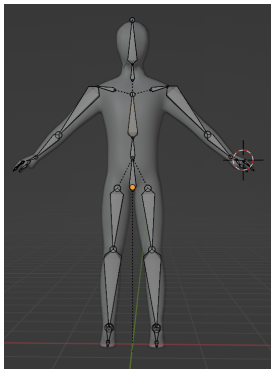

Figure 10: Simple humanoid

To animate the characters on Blender, we first moved the character's bones into key frames that model a regular human walk. We used six key frames and edited the poses to our satisfaction using block chain animation. Then, we applied Blender's Bezier Curves animation option, which smoothed the transitions between key frames.

4 RESULT

The current simulation is still in its infancy and is only comprised of the the core components. The simulation can leverage basic logic to do high level representation of crowed activity. For a more complex simulator with less time investment it would be best to build on existing technology like unity provided meshes, preexisting projects, or external libraries. Building the tools from scratch gave us more granular control, helped us understand the core components that go into a project like this, and allowed us to build a basic working tool.

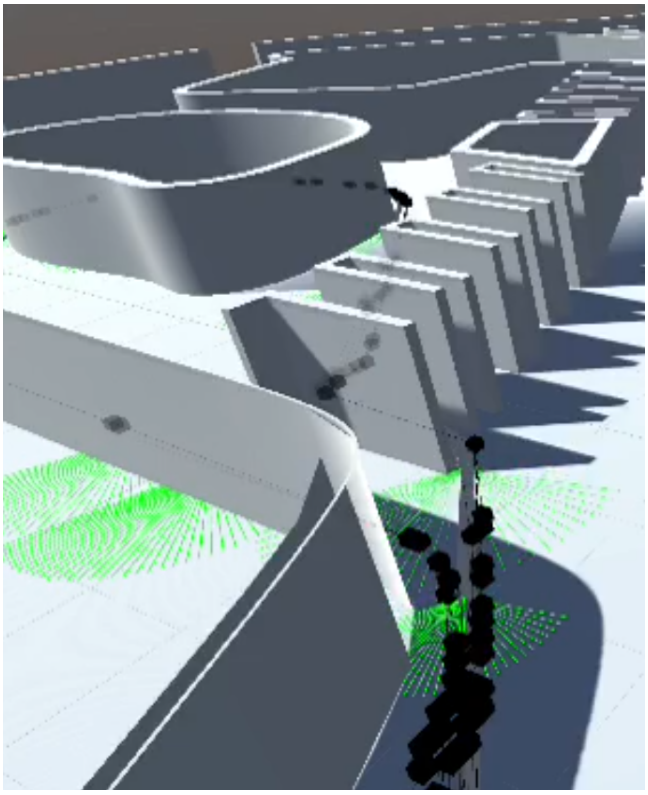

Figure 11: Sample simulation in progress

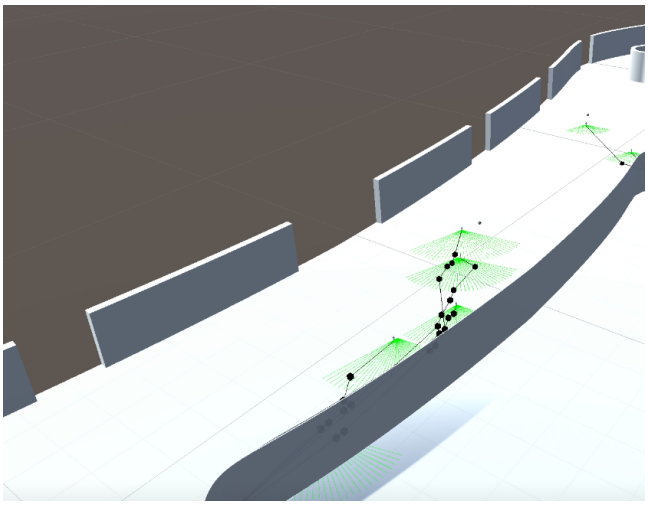

Figure 12: Sample simulation in progress

5 SUMMARY

Circulation has a huge impact on the efficiency of the daily life and more so for emergencies incidents. We propose an agent-based circulation simulation with personalized characters to visualize, evaluate building plans and reversely inform the design decision. We used high level planning and a flocking algorithm to simulate crowd behaviour, adding personalities to individual agents, as well as ray tracing to perform collision detection. Results show the algorithm can simulate complex behaviour within a complex building plan.

However, there are several limitations that we can improve in the future. We only implemented a single floor simulation in 3D. There are still rough-edges in the simulator and some nice to have features missing like industry standard evaluation metrics or a map editor. Adding the key features and smoothing out the rough edges would allow us to integrate the core simulation tech together into a full efficient pipeline that adequately models crowed activity. Future work includes but is not limited to improving models, animation smoothing, adding more complex logic, and integrating with 3d multi-level movement. In the future we would like to include video footage and compare the simulation result with the real world scenario to evaluate and improve the accuracy of the simulation.

ACKNOWLEDGMENTS

We would like to thank Prof. Ren Ng for his wonderful lectures in Computer Graphics and Imaging. We'd also like to acknowledge all the GSIs for their advice for developing this project.

REFERENCES

[1] [n.d.]. Base Character. ([n. d.]). https://free3d.com/3d-model/base-character-ready-to-animate-453899.html
[2] Demeter Dzadik. [n.d.]. Rain Rig © Blender Foundation. ([n. d.]). cloud.blender.org
[3] Kevin Jordao, Panayiotis Charalambous, Marc Christie, Julien Pettré, and Marie-Paule Cani. 2015. Crowd art: density and flow based crowd motion design. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*. 167–176.
[4] Wang J. Xu S. et al. Li, J. 2022. The effect of geometric layout of exit on escape mechanism of crowd. *Building and simulation* 15 (2022), 659–668. https://doi.org/10.1007/s12273-021-0799-2
[5] Nuria Pelechano, Jan M Allbeck, and Norman I Badler. 2007. Controlling individual agents in high-density crowd simulation. (2007).
[6] Daniel Thalmann. 2016. *Crowd Simulation.* Springer International Publishing, Cham, 1–8. https://doi.org/10.1007/978-3-319-08234-9_69-1
[7] Mingliang Xu, Xiaozheng Xie, Pei Lv, Jiangwei Niu, Hua Wang, Chaochao Li, Ruijie Zhu, Zhigang Deng, and Bing Zhou. 2018. Crowd Behavior Simulation with Emotional Contagion in Unexpected Multi-hazard Situations. *CoRR* abs/1801.10000 (2018). arXiv:1801.10000 http://arxiv.org/abs/1801.10000
[8] Ming-Liang Xu, Hao Jiang, and Xiaogang Jin. 2014. Crowd Simulation and Its Applications: Recent Advances. *Journal of Computer Science and Technology* 29 (09 2014), 799–811. https://doi.org/10.1007/s11390-014-1469-y
[9] Shanwen Yang, Tianrui Li, Xun Gong, Bo Peng, and Jie Hu. 2020. A review on crowd simulation and modeling. *Graphical Models* 111 (2020), 101081. https://doi.org/10.1016/j.gmod.2020.101081
[10] H. Yeh, S. Curtis, S. Patil, J. van den Berg, D. Manocha, and M. Lin. 2008. Composite Agents. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Dublin, Ireland) *(SCA '08)*. Eurographics Association, Goslar, DEU, 39–47.
[11] Suiping Zhou, Dan Chen, Wentong Cai, Linbo Luo, Malcolm Yoke Hean Low, Feng Tian, Victor Su-Han Tay, Darren Wee Sze Ong, and Benjamin D Hamilton. 2010. Crowd modeling and simulation technologies. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 20, 4 (2010), 1–35.