Pixels to Play: A Foundation Model for 3D Gameplay

Yuguang Yue, Chris Green, Samuel Hunt, Irakli Salia, Wenzhe Shi, Jonathan J Hunt
Player2
jj@player2.game

***Abstract*—We introduce Pixels2Play-0.1 (P2P0.1), a foundation model that learns to play a wide range of 3D video games with recognizable human-like behavior. Motivated by emerging consumer and developer use cases—AI teammates, controllable NPCs, personalized live-streamers, assistive testers—we argue that an agent must rely on the same pixel stream available to players and generalize to new titles with minimal game-specific engineering. P2P0.1 is trained end-to-end with behavior cloning: labeled demonstrations collected from instrumented human game-play are complemented by unlabeled public videos, to which we impute actions via an inverse-dynamics model. A decoder-only transformer with auto-regressive action output handles the large action space while remaining latency-friendly on a single consumer GPU. We report qualitative results showing competent play across simple Roblox and classic MS-DOS titles, ablations on unlabeled data, and outline the scaling and evaluation steps required to reach expert-level, text-conditioned control.**

I. Introduction

Artificial intelligence (AI) has been applied to game playing since its inception [1]. Human performance in video games correlates with intelligence [2], [3]. Games provide a cheap, safe, and quantifiable environment for evaluating new approaches.

In parallel, large language models (LLMs) such as ChatGPT [4] have ushered general-purpose AI into daily life. Most commercial LLM offerings are now multi-modal visual language models (VLMs), accepting images as input. However, even with latency and cost constraints removed, state-of-the-art VLMs struggle to finish the first level of the 1996 shooter Quake [5].

We are working to close this gap with Pixels2Play 0.1 (P2P0.1), a foundation model trained end-to-end to play any 3D title from raw pixels. Like a novice human, P2P0.1 is expected to perform at a non-trivial level on unseen games without per-game engineering, improving with additional exposure. Crucially, our goal is to make the model text-conditioned. Generating behavior in response to prompts such as "win using only an ax" or "play defensively", allowing a richer human-AI interaction. The level of understanding required for this task is significantly higher than many, particularly reinforcement learning (RL) based approaches, which aim solely to speedrun the game.

[*] Short paper.

A. Uses for such a model

In designing our model, we have benefited from the recent robotics literature [6]–[8], particularly in the area of learning from unlabeled video data. Unlike in robotics, however, where the goal is often to alleviate a boring or dangerous task, humans play video games by choice for recreation. This leads to an obvious question: why would we want an AI to play video games?

We have been using early versions of our model to explore a number of consumer experiences enabled by our work.

- **Gaming companions.** Co-operative titles are often more enjoyable with a friend; When friends are unavailable, our AI can keep the experience social.
- **Adaptive NPCs.** Pixel-driven control frees designers from brittle scripts, enabling richer, emergent interactions without game-specific code.
- **Play-while-you-watch assistants.** Players can let the model handle repetitive sections and intervene when desired - a personalized live streamer on demand.
- **Automated QA.** Text prompts let testers focus the agent on edge cases (e.g. stress testing collision, exploring hidden areas), accelerating bug discovery.

II. Related work

The literature on AI in game play is large; here we focus on recent work on video games.

One focus has been the use of reinforcement learning (RL) to train models to play games. This requires instrumenting the game to extract a reward, such as the score or win/loss of the game. For that reason, this approach is typically limited to playing a single game. Most of the results are based on model-free RL; [9] is a notable exception. Many approaches also substitute engineered state representations for raw pixels, adding additional game-specific engineering. With abundant computation, RL can achieve superhuman play, [10]–[12], although the resulting policies often differ markedly from the human style.

Behavior cloning (BC) reframes control as supervised learning. [13], [14] applied behavior cloning in a single game, while [15] trained a single model in multiple games. [16], [17] have investigated the scaling laws of behavior cloning. [18] used offline RL to generate human-like behavior, but in an engineered state space rather than from pixels.

[19] trained a behavior cloning model to play Minecraft. Most of the training data was "unlabeled" from online video

sources. In order to use these data, this work first trained an inverse dynamics model (IDM) from the labeled data and then used this to impute labels on the unlabeled data. Similar ideas have recently become popular in robotics, where learning from unlabeled videos is now common. Most robotics papers generate *latent* actions: an IDM infers latent controls that are trained to encode information, via a forward dynamics (world) model, useful for predicting the next frame. Policies are first trained on those latent actions and later mapped to real motor commands [6], [20], [21].

III. Methods

A. Policy model

Here we detail our architecture and training procedure for a general game-playing policy P2P0.1.

As in recent work, the core network is a decoder-only transformer [22] (Fig. 1 shows the architecture). Each video frame is first tokenized (see the tokenizer details below) and then fed to the policy transformer; we append a small number of "thinking" tokens that allow extra computation before an action is emitted. Both training and inference run at 20 Hz.

Actions are generated auto-regressively. Previous single game studies collapsed the entire control vector into one categorical prediction [19]. Our many-game setting must accommodate the full keyboard–mouse space, including up to four simultaneous key presses. An autoregressive factorization avoids combinatorial explosion of the action space and supports any distribution over the action space. Continuous inputs (e.g. mouse motion) are discretized, and all heads are trained with cross-entropy loss. The trade-off is higher inference cost, because the network is rolled forward once per sub-action; we mitigate this with standard key-value-caching and the model runs in real-time on a single RTX 5090 GPU.

A conventional causal mask blocks attention from future tokens. That is overly restrictive here because all image tokens arrive together in a single pass. Instead, we apply the causal mask only to the auto-regressive action segment, as illustrated in Fig. 2 (all well as preventing attending to future frames).

Behavior-cloning agents often suffer causal confusion [23]: e.g. keys are often held for multiple frames, and the network may learn to copy the previous action rather than attend to pixels. The offline metrics looked good, but the online gameplay was poor until we masked past actions.

Masking prior actions sacrifices optimality in certain edge cases. For instance, in the game Need for Speed, the player may shift into gear at any point during the pre-race countdown; without seeing earlier actions, the model cannot learn the gradually increasing probability of shifting into gear. Empirically, the policy remains adequate. We expect to enable the action history once larger models and data sets reduce the overfitting risk.

B. Inverse Dynamic Model (IDM)

Unlabeled gameplay videos greatly outnumber curated demonstrations, so we rely on an Inverse Dynamics Model (IDM) to turn those videos into additional training data. Two

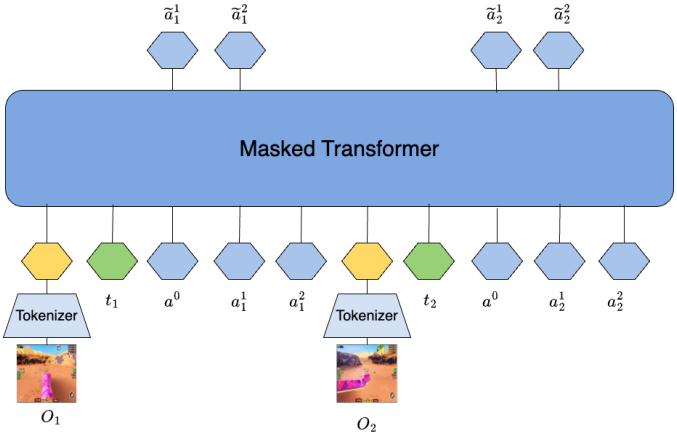

Fig. 1: Architecture of P2P0.1. Each video frame o_i is tokenised and fed to a decoder-only transformer, followed by k learnable "thinking" tokens t_i that grant the model extra computation time. The network then generates the sub-actions a_i^j auto-regressively; the special token a^0 is a learnable embedding that marks the start of the action sequence for step i.

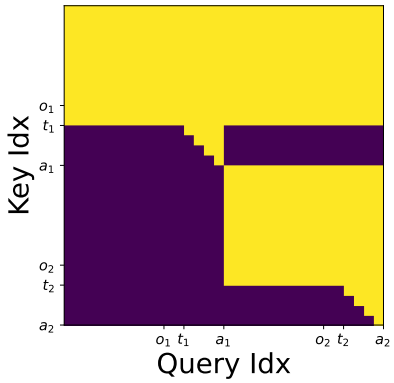

Fig. 2: Attention mask used in our transformer policy (yellow denotes 1 and blue 0). Tick marks show the boundaries of successive inputs. The mask can be read by looking on the x-axis for the query and translating up to see the parts of the key masked out, e.g. tokens in o_1 can attend to any other tokens in o_1 and t_1. The "staircases" are the autoregressive mask while sampling the actions. Finally, you can observe in step 2 that the prior actions in step 1 are masked out.

IDM approaches appear in the literature: a *real-action* model that predicts explicit key/mouse actions [19], and a *latent-action* model that predicts abstract action codes later mapped to real actions [6], [7], [24]. For its simplicity and ability to scale, this work adopts the real-action variant; a direct comparison with latent-action IDMs is left for future study.

Formally, the IDM is a classifier over the action at time t given the surrounding image sequence:

$$\hat{a}_t \sim p_{\text{IDM}}\big(a_t \mid o_1, o_2, \ldots, o_t, \ldots, o_T\big).$$

Our IDM architecture first encodes the frame stack with a

3D convolutional block [19]. The resulting embeddings feed a decoder-only transformer *without* a causal mask, allowing the network to freely attend to past and future frames when inferring the action for each timestep.

Training minimizes cross-entropy between the predicted distribution $\hat{a}_t$ and the ground truth action a_t on a labeled dataset. After convergence, the IDM predicts actions for unlabeled clips, yielding a much larger, imputed-labeled corpus. We then train the policy on the full mix of labeled and imputed-labeled data.

1) Image tokenizers: We have tested a variety of approaches to image tokenizers including using a pre-trained (but with weights unfrozen) convolutional net [13], linear projections of image patches [25], and pre-training a MagVitV2 tokenizer [26] on our dataset (both unlabeled and labeled data). All three deliver broadly similar policy performance. We have found (as in [27]) that using public pre-trained general image tokenizers performed poorly, probably because these tokenizers are typically trained on a large amount of photos, which differ in their visual qualities from games. We also notice that when playing a game the agent must attend to small, fast-moving cues (e.g., the small ball in Blade Ball or the miniature creature in Be a Snake), whereas standard image-recognition models often only capture broader, scene-level context.

2) Data collection: For unlabeled video data, we use public data sources and commercially available VLMs to curate the dataset. We implemented a two-step filtering process. First, an initial filter is applied based on metadata such as the video's title, description, topic, and thumbnail image (when available). This step involves querying a commercial VLM to assess the relevance of the videos to our specified query. Second, the full video content is processed by the VLM to segment and remove non-gameplay scenes, which will not be useful for our model training(e.g. introductions and some visual effects).

For labeled game playing, we use a combination of paid annotators who are requested to play specific games and are exploring the capture of gameplay (with consent) of our product users.

We initially found a significant distributional shift between model training data and inference, which we traced to two factors: 1. During inference, no video compression takes place, but training data (for practical reasons) must be compressed. 2. The image resizing function differed between training (Python) and inference (Rust). We alleviated the distributional shift by using data augmentation to improve robustness, randomizing compression quality during video compression, and using the same resizing function between training and inference.

3) Evaluation: During training, we can easily observe both training and validation loss. However, offline metrics may not correspond to the online performance of the model. A significant challenge for evaluation is that we wish our model to play a large variety of games. Instrumenting even a single game for automated performance evaluation is time-consuming. For this reason, currently, we are primarily limited to a qualitative evaluation of the model's performance. This is an area that we intend to develop further.

Fig. 3: Examples of Roblox and MS-DOS games P2P0.1 is currently capable of playing. Videos of the policy in action can be viewed at the accompanying blog post https://blog.player2.game/p/pixels-to-play-one-model-any-game. We intend to make the model available for interactive demos at the conference.

IV. Experiments

A. General game playing

Currently, we have focused on simple Roblox games along with some older MS-DOS titles. The Roblox platform has the advantage that, as it reduces the barriers to game design, it has a very wide variety of games. We are also learning MS-DOS games as part of a goal to use automatic evaluation in the future. In all games, we capture training data and evaluate directly on end-user computers with no instrumentation or modification to the games.

As discussed above, instrumented evaluation is an area of active work. Qualitatively, we find that P2P0.1 is capable of playing games currently at the level of an novice human (Fig. 3, that is, it can play most games we trained on, but a skilled human player will outperform the model.

B. Unlabeled data helps generalization

This experiment measures how additional unlabeled data affect generalization. The datasets follow Section III-B2. We train three variants:

- Full-Label, using 100 % of the training labeled data;
- Limited-Label, using 10 % of the training labeled data;
- Imputed-Label, using the same 10 % training labeled data plus unlabeled data being imputed by IDM.

Hyper-parameters are identical across runs, and Imputed-Label is matched to Full-Label for the total number of training frames. All models are validated on the same held-out labeled set.

Figures 4 and 5 show the learning curves. Note that we stopped the run of Limited-Label training once we clearly observed overfitting to reduce the cost of resources. Limited-Label overfits rapidly after around 30 epochs through the limited dataset, while Full-Label and Imputed-Label continue to improve throughout training. The best validation perplexities are 1.40 (Limited-Label) and 1.08 (Augmented-Label), a reduction 22% attributable to the imputed-label data. A residual gap between full-label and enhanced-label highlights the quality difference and remaining domain shift between true labels and imputed labels.

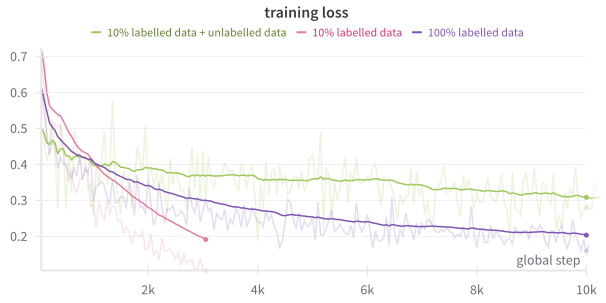

Fig. 4: Training loss curve across models with different data mixture

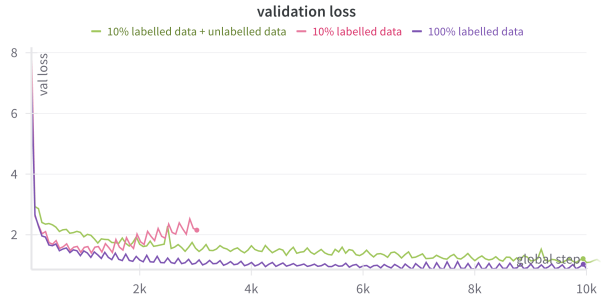

Fig. 5: Validation loss curve across models with different data mixture

V. Discussion

This paper reports on initial progress toward a foundation model that produces human-like behavior directly from pixels in 3D video games. We have also discussed early user-facing prototypes—AI companions, smarter NPCs, and play-assist tools—that both validate the approach and create new streams of real gameplay data for future training.

Currently, P2P0.1 handles a range of relatively simple 3D titles. The ongoing work focuses on two main fronts. First, we continue to iterate on architecture and scaling, enlarging both the labeled and unlabeled corpora and increasing model capacity. Second, we are extending the temporal window so that the agent can reason over much longer histories, a prerequisite for competent play in more complex games.

References

[1] A. M. Turing, "Digital computers applied to games," *Faster than thought*, 1953.

[2] A. V. Kokkinakis, P. I. Cowling, A. Drachen, and A. R. Wade, "Exploring the relationship between video game expertise and fluid intelligence," *PloS one*, vol. 12, no. 11, p. e0186621, 2017.

[3] H. Peters, A. Kyngdon, and D. Stillwell, "Construction and validation of a game-based intelligence assessment in minecraft," *Computers in Human Behavior*, vol. 119, p. 106701, 2021.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[5] A. L. Zhang, T. L. Griffiths, K. R. Narasimhan, and O. Press, "Videogamebench: Can vision-language models complete popular video games?" 2025. [Online]. Available: https://arxiv.org/abs/2505.18134

[6] S. Ye, J. Jang, B. Jeon, S. Joo, J. Yang, B. Peng, A. Mandlekar, R. Tan, Y.-W. Chao, B. Y. Lin *et al.*, "Latent action pretraining from videos," *arXiv preprint arXiv:2410.11758*, 2024.

[7] J. B. Nvidia, F. Castaneda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *ArXiv, abs/2503.14734*, 2025.

[8] N. Agarwal, A. Ali, M. Bala, Y. Balaji, E. Barker, T. Cai, P. Chattopadhyay, Y. Chen, Y. Cui, Y. Ding *et al.*, "Cosmos world foundation model platform for physical ai," *arXiv preprint arXiv:2501.03575*, 2025.

[9] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse control tasks through world models," *Nature*, pp. 1–7, 2025.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[11] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[12] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[13] T. Pearce and J. Zhu, "Counter-strike deathmatch with large-scale behavioural cloning," in *2022 IEEE Conference on Games (CoG)*. IEEE, 2022, pp. 104–111.

[14] A. Kanervisto, D. Bignell, L. Y. Wen, M. Grayson, R. Georgescu, S. Valcarcel Macua, S. Z. Tan, T. Rashid, T. Pearce, Y. Cao *et al.*, "World and human action models towards gameplay ideation," *Nature*, vol. 638, no. 8051, pp. 656–663, 2025.

[15] M. A. Raad, A. Ahuja, C. Barros, F. Besse, A. Bolt, A. Bolton, B. Brownfield, G. Buttimore, M. Cant, S. Chakera *et al.*, "Scaling instructable agents across many simulated worlds," *arXiv preprint arXiv:2404.10179*, 2024.

[16] J. Tuyls, D. Madeka, K. Torkkola, D. Foster, K. Narasimhan, and S. Kakade, "Scaling laws for imitation learning in single-agent games," *arXiv preprint arXiv:2307.09423*, 2023.

[17] T. Pearce, T. Rashid, D. Bignell, R. Georgescu, S. Devlin, and K. Hofmann, "Scaling laws for pre-training agents and world models," *arXiv preprint arXiv:2411.04434*, 2024.

[18] A. R. Farhang, B. Mulcahy, D. Holden, I. Matthews, and Y. Yue, "Humanlike behavior in a third-person shooter with imitation learning," in *2024 IEEE Conference on Games (CoG)*. IEEE, 2024, pp. 1–4.

[19] B. Baker, I. Akkaya, P. Zhokov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, and J. Clune, "Video pretraining (vpt): Learning to act by watching unlabeled online videos," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 639–24 654, 2022.

[20] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[21] A. Liang, P. Czempin, M. Hong, Y. Zhou, E. Biyik, and S. Tu, "Clam: Continuous latent action models for robot learning from unlabeled demonstrations," *arXiv preprint arXiv:2505.04999*, 2025.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[23] P. De Haan, D. Jayaraman, and S. Levine, "Causal confusion in imitation learning," *Advances in neural information processing systems*, vol. 32, 2019.

[24] D. Schmidt and M. Jiang, "Learning to act without actions," *arXiv preprint arXiv:2312.10812*, 2023.

[25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[26] L. Yu, J. Lezama, N. B. Gundavarapu, L. Versari, K. Sohn, D. Minnen, Y. Cheng, V. Birodkar, A. Gupta, X. Gu *et al.*, "Language model beats diffusion–tokenizer is key to visual generation," *arXiv preprint arXiv:2310.05737*, 2023.

[27] L. Schäfer, L. Jones, A. Kanervisto, Y. Cao, T. Rashid, R. Georgescu, D. Bignell, S. Sen, A. T. Gavito, and S. Devlin, "Visual encoders for data-efficient imitation learning in modern video games," *arXiv preprint arXiv:2312.02312*, 2023.