Promptable Game Models: Text-guided Game Simulation via Masked Diffusion Models

WILLI MENAPACE, University of Trento, Italy
ALIAKSANDR SIAROHIN, Snap Inc., USA
STÉPHANE LATHUILIÈRE, LTCI, Télécom Paris, Institut Polytechnique de Paris, France
PANOS ACHLIOPTAS, Snap Inc., USA
VLADISLAV GOLYANIK, MPI for Informatics, SIC, Germany
SERGEY TULYAKOV, Snap Inc., USA
ELISA RICCI, University of Trento, Fondazione Bruno Kessler, Italy

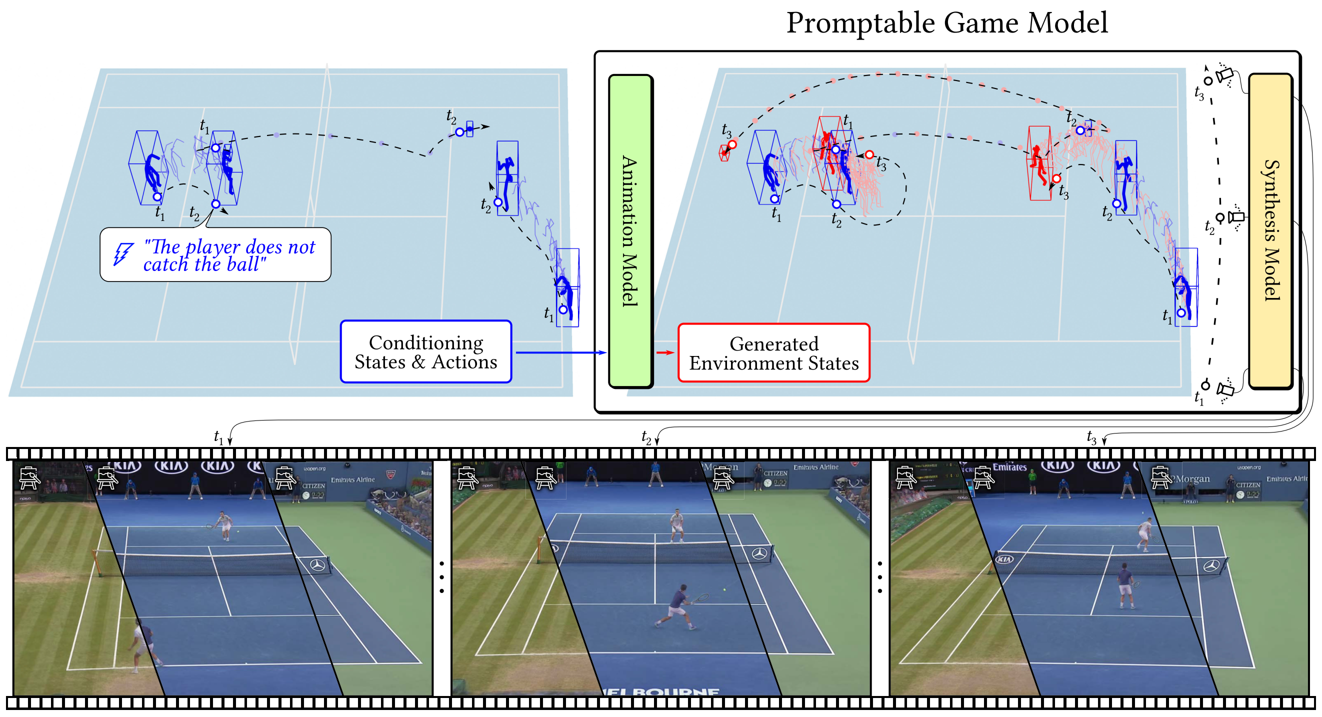

Promptable Game Model
"The player does not catch the ball"
Conditioning States & Actions
Animation Model
Generated Environment States
Synthesis Model
t_1
t_2
t_3

Fig. 1. We propose Promptable Game Models (PGMs), controllable models of games that are learned from annotated videos. Our PGM enables the generation of videos using prompts, a wide spectrum of conditioning signals such as player poses, object locations, and detailed textual actions (see) indicating what each player should do. Our *Animation Model* uses this information to generate future, past, or interpolated environment states according to the learned game dynamics. At this stage, the model is able to perform complex action reasoning such as generating a winning shot if the action "the [other] player does not catch the ball" is specified, as shown in the figure. To accomplish this goal, the model decides that the bottom player should hit the ball with a "lob" shot, sending the ball high above the opponent, who is unable to catch it. Our model renders the scene from a user-defined viewpoint (see) using a *Synthesis Model* where the style of the scene (see) can be controlled explicitly.

Neural video game simulators emerged as powerful tools to generate and edit videos. Their idea is to represent games as the evolution of an environment's state driven by the actions of its agents. While such a paradigm

W. Menapace's work performed while the author was an intern at Snap Inc.
This work was partially supported by the EU HEU AI4TRUST (101070190) project.
Authors' addresses: W. Menapace, University of Trento, Italy; e-mail: willi.menapace@gmail.com; A. Siarohin, P. Achlioptas, and S. Tulyakov, Snap Inc., USA; e-mails: asiarohin@snapchat.com, pachlioptas@gmail.com, stulyakov@snapchat.com; S. Lathuilière, LTCI, Télécom Paris, Institut Polytechnique de Paris, France; e-mail: stephane.lathuiliere@telecom-paris.fr; V. Golyanik, MPI for Informatics, SIC, Germany; e-mail: golyanik@mpi-inf.mpg.de; E. Ricci, University of Trento, Fondazione Bruno Kessler, Italy; e-mail: e.ricci@unitn.it.

0730-0301/2024/01-ART17 $15.00
https://doi.org/10.1145/3635705

enables users to *play* a game action-by-action, its rigidity precludes more semantic forms of control. To overcome this limitation, we augment game models with *prompts* specified as a set of *natural language* actions and *desired states*. The result—a Promptable Game Model (PGM)—makes it possible for a user to *play* the game by prompting it with high- and low-level action sequences. Most captivatingly, our PGM unlocks the *director's mode*, where the game is played by specifying goals for the agents in the form of a prompt. This requires learning "game AI," encapsulated by our animation model, to navigate the scene using high-level constraints, play against an adversary, and devise a strategy to win a point. To render the resulting state, we use a compositional NeRF representation encapsulated in our synthesis model. To foster future research, we present newly collected, annotated and calibrated Tennis and Minecraft datasets. Our method significantly outperforms existing neural video game simulators in terms of rendering quality and unlocks applications beyond the capabilities of the current state-of-the-art. Our framework, data, and models are available at snap-research.github.io/promptable-game-models.

CCS Concepts: • **Computing methodologies** → **Rendering**; **Animation;**

Additional Key Words and Phrases: Neural radiance fields, diffusion models, human motion generation, language modeling

ACM Reference format:
Willi Menapace, Aliaksandr Siarohin, Stéphane Lathuilière, Panos Achlioptas, Vladislav Golyanik, Sergey Tulyakov, and Elisa Ricci. 2024. Promptable Game Models: Text-guided Game Simulation via Masked Diffusion Models. *ACM Trans. Graph.* 43, 2, Article 17 (January 2024), 16 pages.
https://doi.org/10.1145/3635705

1 INTRODUCTION

Recent video generation methods, thanks to their training on extensive web-scale datasets [Schuhmann et al. 2022], exhibit a remarkable capacity for generating a vast amount of different concepts and scenes [Blattmann et al. 2023; Ho et al. 2022a; Singer et al. 2022]. Despite this, their generic nature hinders their comprehension of the dynamics of the modeled scenes. When generating or editing videos of a game, such as a tennis match, this limitation impedes their ability to attain precise control of the player movements or to devise optimal strategies to reach desired states of the game, such as victory over the opponent.

Neural video game simulators, a growing category of video generation methods, make an important step in this direction by focusing on modeling the dynamics of an environment, often a sports or computer game, with high fidelity and degree of control, and show that annotated videos can be used to learn to generate videos interactively [Davtyan and Favaro 2022; Huang et al. 2022; Kim et al. 2021, 2020; Menapace et al. 2021] and build 3D environments where agents can be controlled through a set of discrete actions [Menapace et al. 2022]. However, when applied to complex or real-world environments, these works present several limitations: They do not accurately model the game's dynamics, do not model physical interactions of objects in 3D space, do not learn precise controls, do not allow for high-level goal-driven control of the game flow, and, finally, do not model intelligent behavior of the agents, a capability often referred to as "game AI."

In this work, we overcome these limitations by introducing game models trained on a set of annotated videos that support complex prompts. Due to the versatility of the applications enabled by diverse prompting methods (see Section 4), we call them **Promptable Game Models (PGMs)**. More formally, we define PGMs as those models supporting a core set of game modeling and prompting functions including rendering from a controllable viewpoint, modeling of game's dynamics, precise character control, high-level goal-driven control of the game, and game AI. Making a first step towards the realization of such models, we propose a framework that supports these characteristics.

To overcome the limitations of Davtyan and Favaro [2022], Huang et al. [2022], Kim et al. 2021, 2020, and Menapace et al. 2021, 2022, not only do we model the *states* of an environment, but we also consider detailed textual representations of the actions taking place in it. We argue that training on user commentaries describing detailed actions of a game greatly facilitates learning the dynamics of the game and game AI—important parts of PGMs—and that such commentaries are a key component in enabling a series of important model capabilities related to precise character control and high-level goal-driven control of the game flow.

In its simplest form, for games like tennis, this enables controlling each player in a precise manner with instructions such as "*hit the ball with a backhand and send it to the right service box*." Moreover, language enables users to take the *director's mode* and prompt the model with high-level game-specific scenarios or scripts, specified by means of *natural language* and *desired states of the environment*. As an example, given desired starting and ending states, our promptable game model can devise in-between scenarios that led to the observed outcome. Most interestingly, as shown in Figure 1, given the initial states of a real tennis video in which a player lost a point, our model prompted by the command "*the [other] player does not catch the ball*" can perform the necessary action to win the point.

Broadly speaking, a game maintains states of its environments [Curtis et al. 2022; Stanton et al. 2016; Starke et al. 2019], renders them using a controllable camera, and evolves them according to user commands, actions of non-playable characters controlled by the game AI, and the game's dynamics. Our framework follows this high-level structure highlighted in Figure 1. Our synthesis model maintains a state for every object and agent included in the game and renders them in the image space using the compositional NeRF of Menapace et al. [2022] followed by a learnable enhancer for superior rendering quality. To model the dynamics of games and game AI that determine the evolution of the environment states, we introduce an animation model. Specifically, inspired by Han et al. [2022], we train a *non-autoregressive text-conditioned* diffusion model that leverages masked sequence modeling to express the conditioning signals corresponding to a prompt. In particular, we show that using text labels describing actions happening in a game is instrumental in learning such capabilities. While certain prior work [Kim et al. 2021, 2020; Menapace et al. 2021, 2022] explored maintaining and rendering states of games, we are not aware of any generative method that attempts to enable precise control, modeling sophisticated goal-driven game dynamics, and learning game AI to the extent explored in this article.

The task of playing games and manipulating videos in the *director's mode* has not been previously introduced in the literature. With this work, we attempt to introduce the task and set up a solid framework for future research. To do that, we collected two monocular video datasets. The first one is the Minecraft dataset containing 1.2 hours of videos, depicting a player moving in a complex environment. The second is a large-scale real-world dataset with 15.5 hours of high-resolution professional tennis matches. For each frame in these datasets, we provide accurate camera calibration, 3D player poses, ball localization, and, most importantly, diverse and rich text descriptions of the actions performed by each player in each frame.

In summary, our work brings the following contributions:

— A framework for the creation of Promptable Game Models. It supports detailed offline rendering of high-resolution, high-frame rate videos of scenes with articulated objects from controllable viewpoints. It can generate actions specified by detailed text prompts, model opponents, and perform goal-driven generation of complex action sequences. As far as we are aware, no existing work provides this set of capabilities under comparable data assumptions.
— A synthesis model, based on a compositional NeRF backed by an efficient plane- and voxel-based object representation that operates without upsampling. With respect to the upsampler-based approach of Menapace et al. [2022], it doubles the output resolution, can synthesize small objects, and does not present checkerboard upsampling artifacts.
— An animation model, based on a text-conditioned diffusion model with a masked training procedure, which is key to supporting complex game dynamics, object interactions, game AI, and understanding detailed actions. It unlocks applications currently out of reach of state-of-the-art neural video game simulators (see Section 4).
— A large-scale 15 h Tennis and a 1 h Minecraft video datasets with camera calibration, 3D player poses, 3D ball localization, and detailed text captions.

2 RELATED WORK

Our Promptable Game Model relates to neural game simulation literature, game engines, character animation, neural rendering, sequential data generation, and text-based generation. We review the most recent related works in this section.

2.1 Neural Video Game Simulation

In the past few years, video game simulation using deep neural networks has emerged as a new research trend [Davtyan and Favaro 2022; Huang et al. 2022; Kim et al. 2021, 2020; Menapace et al. 2021, 2022]. The objective is to train a neural network to synthesize videos based on a specific type of prompt: a sequence of actions provided at every timestep.

This problem was first addressed using training videos annotated with the corresponding action labels at each timestep [Chiappa et al. 2017; Kim et al. 2020; Oh et al. 2015]. They consider a discrete action representation that is difficult to define *a priori* for real-world environments. More recently, Kim et al. [2021] proposed a framework that uses a continuous action representation to model real-world driving scenarios. Devising a good continuous action representation for an environment, however, is complex. To avoid this complexity, Menapace et al. Menapace et al. [2021, 2022], propose to learn a discrete action representation. Huang et al. [2022] expand on this idea by modeling actions as a learned set of geometric transformations, while Davtyan and Favaro [2022] represent actions by separating them into a global shift component and a local discrete action component.

Differently from our PGM, previous works perform generation in an autoregressive manner, conditioned on the actions and, therefore, are unable to answer prompts entailing constraint- or goal-driven generation for which non-sequential conditioning is necessary. We find the proposed text-based action representation and masked training procedure to be crucial to unlocking such applications.

Among these works, *Playable environments* [Menapace et al. 2022] is the most closely related to ours. Rather than employing a 2D model, they use a NeRF-based renderer [Mildenhall et al. 2020] that enables them to represent complex 3D scenes. We follow this high-level design but introduce a more efficient plane- and voxel-based NeRF representation that enables the rendering of outputs at double the original resolution without the use of upsampling modules, which we found to be the cause of checkerboard artifacts, failures in rendering of small objects, and to be prone to failure when training at higher resolutions. In addition, the employed discrete action representation shows limitations in complex scenarios such as tennis, where it is only able to capture the main movement directions of the players and does not model actions such as ball hitting. In contrast, we employ a text action representation that specifies actions at a fine level of granularity (i.e., which particular ball-hitting action is being performed and where the ball is sent), while remaining interpretable and intuitive for the user. Last, we replace the adversarially trained LSTM animation module with a more capable masked diffusion transformer.

2.2 Game Engines

Game engines brought a revolution to game development by providing extensible and reusable software that can be employed to create a wide range of game models [Gregory 2018]. Nowadays, a range of game engines exists (*Unity*, *Unreal*, *id Tech*, *Source*, *CRYENGINE*, *Frostbite*, *RAGE*) and have grown to become vast software ecosystems. Modern game engines are organized into components including a rendering engine [Müller et al. 2020], a resource manager, a module for physics and collision, an animation manager, and, importantly, a gameplay foundation system that models the game rules and encapsulates game AI functionalities [Gregory 2018]. The presence of these components, coupled with the labor of a range of trained experts including software engineers, artists (animators, 3D modelers, texture and lighting artists), and game developers, enables the construction of sophisticated game models supporting low-level character control and scripted agent behavior. We show that monocular videos annotated with a fraction of the effort (see Supplement I.1) can be used to learn models of games that support answering challenging prompts related to agent intelligence, a capability difficult to achieve through scripted agent behavior.

2.3 Character Animation

Character animation is a long-standing problem in computer graphics. Several recent methods have been proposed that produce high-quality animations. Holden et al. [2020] propose a learnable version of Motion Matching [Büttner and Clavet 2015] that formulates character animation as retrieval of the closest motion from a motion database and supports interaction with other characters or objects. Other approaches model the evolution of characters using time series models conditioned on the preceding state and control signals [Holden et al. 2017; Lee et al. 2018; Starke et al. 2019, 2020]. Starke et al. [2019] propose a model based on a mixture of experts that controls character locomotion and object interactions; in a follow-up work [Starke et al. 2020] they introduce local motion phases to model complex character motions and interaction with a second character.

To produce high-quality animations the methods rely on difficult-to-acquire motion capture data enriched with contact information [Holden et al. 2020; Starke et al. 2019, 2020], motion phases [Starke et al. 2019], or engineered action labels [Starke et al. 2019, 2020]. Additionally, handcrafted dataset-specific feature representations and mappings from user controls to such representations are often leveraged, and additional knowledge is injected through postprocessing steps such as inverse kinematics or external physics models. While these assumptions promote high-quality outputs, they come at a significant effort. In contrast, our method sidesteps these requirements by not using motion capture and basing user control on natural language that is cheaper to acquire (see Supplement I.1) and does not require manual engineering. Finally, character animation methods support limited goal-driven control such as interacting with a specific object while avoiding collisions [Starke et al. 2019]. In contrast, our method models complex game AI tasks such as modeling strategies to defeat the opponent, which are instrumental in answering complex user prompts.

2.4 Neural Rendering

Neural rendering was recently revolutionized by the advent of NeRF [Mildenhall et al. 2020]. Several modifications of the NeRF framework were proposed to model deformable objects [Li et al. 2022; Park et al. 2021a, b; Tretschk et al. 2021; Weng et al. 2022] and decomposed scene representations [Kundu et al. 2022; Menapace et al. 2022; Müller et al. 2022; Niemeyer and Geiger 2021; Ost et al. 2021]. In addition, several works improved the efficiency of the original MLP representation of the radiance field [Mildenhall et al. 2020] by employing octrees [Martel et al. 2021; Yu et al. 2021], voxel grids [Fridovich-Keil et al. 2022], triplanes [Chan et al. 2022], hash tables [Müller et al. 2022], or factorized representations [Chen et al. 2022].

Our framework is most related to that of Weng et al. [2022], since we model player deformations using an articulated 3D prior and **linear blend skinning (LBS)** [Lewis et al. 2000]. Differently from them, however, we consider scenes with multiple players and apply our method to articulated objects with varied structures for their kinematic trees. While similar to the rendering framework of Menapace et al. [2022], our framework does not adopt computationally inefficient MLP representations, using voxel [Fridovich-Keil et al. 2022] or plane representations instead, thus does not rely on upsampler networks.

2.5 Sequential Data Generation with Diffusion Models

In prior work, sequential data generation was mainly addressed with auto-regressive formulations combined with adversarial [Kwon and Park 2019] or variational [Babaeizadeh et al. 2018; Fortuin et al. 2020] generative models. Recently, diffusion models have emerged as a promising solution to this problem, leading to impressive results in multiple applications such as audio [Chen et al. 2021; Kong et al. 2020; Lam et al. 2022; Leng et al. 2022] and video synthesis [Blattmann et al. 2023; Ho et al. 2022a, b; Singer et al. 2022], language modeling [Dieleman et al. 2022], and human motion synthesis [Dabral et al. 2023; Zhang et al. 2022]. Following this methodological direction [Tashiro et al. 2021], introduce a score-based diffusion model for imputing missing values in time series. They introduce a training procedure based on masks that simulate missing data. This approach motivates our choice of a similar masking strategy to model the conditions entailed by the given prompt and generate the unknown environment states. In this work, we show that mask-based training is highly effective in modeling geometric properties together with textual data modalities.

2.6 Text-based Generation

In recent years, we have witnessed the emergence of works on the problem of text-based generation. Several works address the problem of generating images [Ramesh et al. 2022, 2021; Rombach et al. 2021; Saharia et al. 2022] and videos with arbitrary content [Ho et al. 2022a, b; Hong et al. 2022; Singer et al. 2022], and arbitrary 3D shapes [Achlioptas et al. 2023; Jain et al. 2022; Lin et al. 2023].

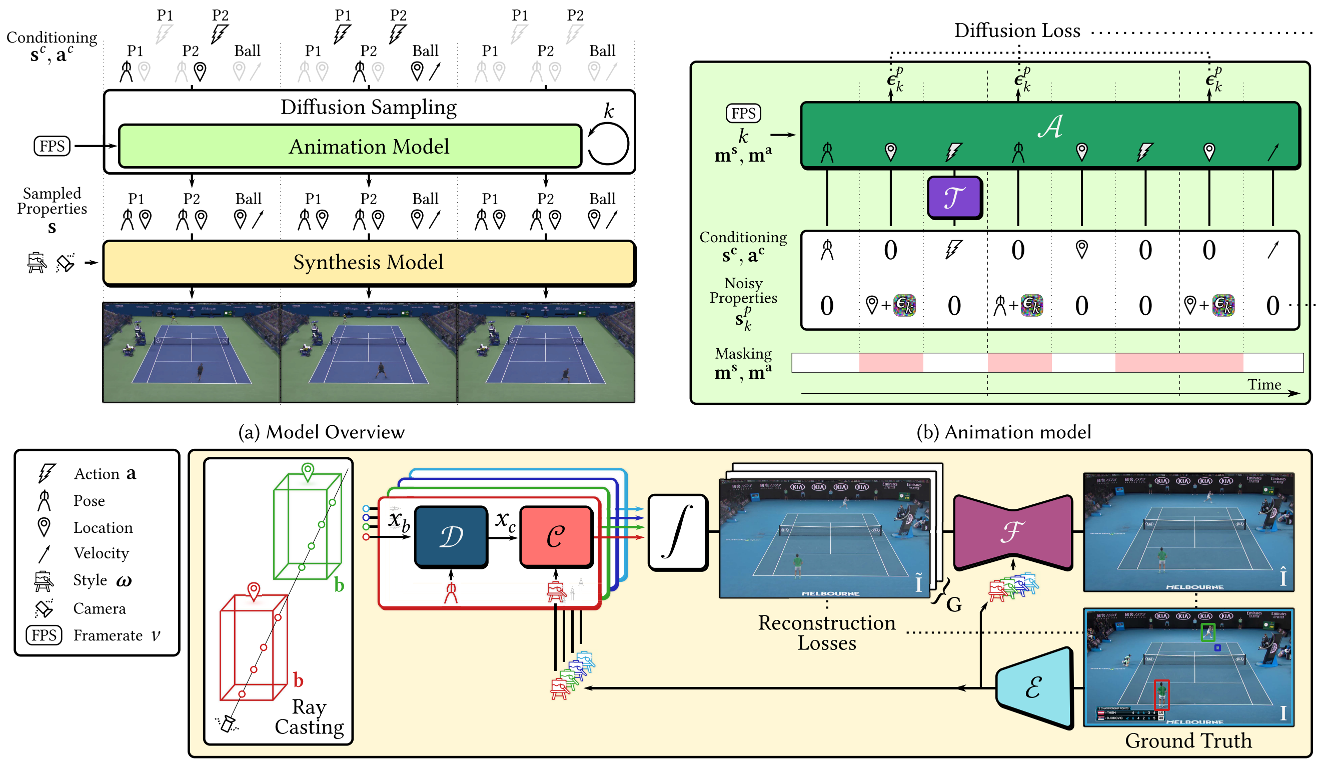
Han et al. [2022] introduced a video generation framework that can incorporate various conditioning modalities in addition to text, such as segmentation masks or partially occluded images. Their approach employs a frozen RoBERTa [Liu et al. 2020] language model and a sequence masking technique. Fu et al. [2023] propose an analogous framework. Our animation framework employs a similar masking strategy, but we model text conditioning at each timestep in the sequence, use diffusion models that operate on continuous rather than discrete data, and generate scenes that can be rendered from arbitrary viewpoints.

More relevant to our work, several papers introduced models to generate human motion sequences from text [Athanasiou et al. 2022; Tevet et al. 2022]. Recently, diffusion models have shown strong performance on this task [Dabral et al. 2023; Zhang et al. 2022]. In these works, sequences of human poses are generated by a diffusion model conditioned on the output of a frozen CLIP text encoder. It is worth noting that these prior works model only a single human, while our framework supports multiple human agents and objects and models their interactions with the environment.

3 METHOD

This section introduces our framework for the creation of *Promptable Game Models* that allows the user to perform a range of dynamic scene editing tasks, formulated as a set of conditioning *prompts*.

We divide our PGM into two modules: a *synthesis model* and an *animation model*. The synthesis model generates an image given the representation of the environment state. The animation model, instead, aims at modeling the game's dynamics, with player actions and interactions, in the high-level space of the environment states. Actions are modeled as text, which is an expressive, yet intuitive form of control for a wide range of tasks. The overview of our framework is provided in Figure 2(a).



(c) Synthesis model

Fig. 2. (a) Overview of our framework. The *animation model* produces states $\mathbf{s}$ based on user-provided conditioning signals, or *prompts*, $\mathbf{s}^c$, $\mathbf{a}^c$ that are rendered by the *synthesis model*. (b) The diffusion-based animation model predicts noise $\boldsymbol{\epsilon}_k$ applied to the noisy states $\mathbf{s}_k^p$ conditioned on known states $\mathbf{s}^c$ and actions $\mathbf{a}^c$ with the respective masks $\mathbf{m}^s$, $\mathbf{m}^a$, diffusion step k and framerate ν. The *text encoder* $\mathcal{T}$ produces embedding for the textual actions, while the *temporal model* $\mathcal{A}$ performs noise prediction. (c) The synthesis model renders the current state using a composition of neural radiance fields, one for each object. A *style encoder* $\mathcal{E}$ extracts the appearance $\boldsymbol{\omega}$ of each object. Each object is represented in its canonical pose by C, and deformations of articulated objects are modeled by the *deformation model* $\mathcal{D}$. After integration and composition, the feature grid $\mathbf{G}$ is rendered to the final image using the *feature enhancer* $\mathcal{F}$.

In more detail, our model defines the state of the entire environment as the combination of all individual object states. Consequently, each individual state is the set of the object properties such as the position of each object in the scene, their appearance, or their pose. Formally, the environment state at time t can be represented by $\mathbf{s}_t \in \mathbb{S} = (\mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_P})$, P properties of variable length n_i defined as the union of the properties of each object. This state representation captures all variable aspects of each object in the environment, thus it can be used by the synthesis model to generate the scene.

However, the animation model predicts the evolution of an environment in time, which is represented by the sequence of its states $\{\mathbf{s}_1, \mathbf{s}_2, \dots \mathbf{s}_T\} = \mathbf{s} \in \mathbb{S}^T$, where T is the length of the sequence. The model provides control over sequence generation with the help of user-defined conditioning signals, or prompts, that can take two forms: explicit state manipulation and high-level text-based editing. With respect to the former, the user could change the position of the tennis ball at timestep t, and the model would automatically adapt the position of the ball in other nearby states. As far as the latter is concerned, users could provide high-level text-based values of actions such as "*The player takes several steps to the right and hits the ball with a backhand*" and the model would generate the corresponding sequence of states (see Figure 3). These generic actions in the form of text are central to enabling high-level, yet fine-grained control over the evolution of the environment. To train our framework, we assume a dataset of camera-calibrated videos, where each video frame is annotated with the corresponding states $\mathbf{s}$ and actions $\mathbf{a}$.

3.1 Synthesis Model

In this section, we describe the synthesis model that renders states from controllable viewpoints (see Figure 2(c)). We build our model based on a compositional NeRF [Menapace et al. 2022] framework that enables explicit control over the camera and represents a scene as a composition of different, independent objects. Thanks to the independent representation of objects, each object property is directly linked to an aspect of the respective object and can thus be easily controlled and manipulated. The compositional NeRF framework allows different, specialized NeRF architectures to be used for each object based on its type. To further improve quality, rather than directly rendering RGB images with the NeRF models, we render features and make use of a feature enhancer CNN to produce the RGB output. To represent objects with different appearances, we condition the NeRF and enhancer models on the style codes extracted with a dedicated style encoder [Menapace et al. 2022]. Our model is trained using reconstruction as the main guiding signal.

In Sections 3.1.1–3.1.6, we illustrate the main components of the synthesis module, and in Section 3.1.7, we describe the training procedure.

3.1.1 *Scene Composition with NeRFs.*

Neural radiance fields represent a scene as a radiance field, a 5D function parametrized as a neural network mapping the current position $\mathbf{x}$ and viewing direction $\mathbf{d}$ to density σ and radiance $\mathbf{c}$.

To allow controllable generation of complex scenes, we adopt a compositional strategy where each object in the scene is modeled with a dedicated NeRF model [Menapace et al. 2022; Müller et al. 2022; Xu et al. 2022]. The scene is rendered by sampling points independently for each object and querying the respective object radiance field C_i. The results for all objects are then merged and sorted by distance from the camera before being integrated.

All objects are assumed to be described by a set of properties whose structure depends on the type of object, e.g., a player, the ball, the background. We consider the following properties:

- *Object location.* Each object is contained within an axis-aligned bounding box $\mathbf{b}_i^{3D}$, which is defined by size and position. In the case of the ball, we additionally consider its velocity to model blur effects (Section 3.1.6).
- *Object style.* All objects have an appearance that may vary in different sequences, thus we introduce a style code $\boldsymbol{\omega}_i$ as an additional property for all objects. Since it is difficult to define such style information *a priori*, we assume it to be a latent variable and learn it jointly during training.
- *Object pose.* Articulated objects such as humans require additional properties to model varying poses. We model the deformation of articulated objects as a kinematic tree with J_i joints and consider as object properties the rotation $\mathbf{R}$ and translation $\mathbf{tr}$ parameters associated with each joint (Section 3.1.4).

From now on, we drop the object index i to simplify notation.

3.1.2 *Style Encoder.*

Representing the appearance of each object is challenging, since it changes based on the type of object and illumination conditions. We treat the style $\boldsymbol{\omega}$ for each object as a latent variable that we regress using a convolutional style encoder $\mathcal{E}$. Given the current video frame $\mathbf{I}$ with O objects, we compute 2D bounding boxes $\mathbf{b}^{2D}$ for each object. First, a set of residual blocks is used to extract frame features that are later cropped around each object according to $\mathbf{b}^{2D}$ using RoI pooling [Girshick et al. 2013]. Later, a series of convolutional layers with a final projection is used to predict the style code $\boldsymbol{\omega}$ from the cropped feature maps.

3.1.3 *Volume Modeling for Efficient Sampling.*

Radiance fields are commonly parametrized using MLPs [Mildenhall et al. 2020] but such representation requires a separate MLP evaluation for each sampled point, making it computationally challenging to train high-resolution models. To overcome such issue, we model the radiance field C of each object in a canonical space using two alternative parametrizations.

For three-dimensional objects, we make use of a voxel grid parametrization [Fridovich-Keil et al. 2022; Weng et al. 2022]. Starting from a fixed noise tensor $\mathbf{V'} \in \mathbb{R}^{F' \times H'_V \times W'_V \times D'_V}$, a series of 3D convolutions produces a voxel $\mathbf{V} \in \mathbb{R}^{F+1 \times H_V \times W_V \times D_V}$ containing the features and density associated to each point in the bounded space. Here, F' and F represent the number of features, while H_V, W_V, and D_V represent the size of the voxel. Given a point in the object canonical space $\mathbf{x}_c$, the associated features and density σ are retrieved using trilinear sampling on $\mathbf{V}$. To model the different appearance of each object, we adopt a small MLP conditioned on the style $\boldsymbol{\omega}$ to produce a stylized feature with the help of weight demodulation [Karras et al. 2020].

For two-dimensional objects such as planar scene elements, we make use of a similar parametrization where a fixed 2D noise tensor $\mathbf{P'} \in \mathbb{R}^{F' \times H'_P \times W'_P}$ is mapped to a plane of features $\mathbf{P} \in \mathbb{R}^{F \times H_P \times W_P}$ using a series of 2D convolutions. Given a ray r, we compute the intersection point $\mathbf{x}$ between the plane and the ray that is used to sample $\mathbf{P}$ using bilinear sampling. Similarly to the voxel case, a small MLP is used to model object appearance according to $\boldsymbol{\omega}$. We assume planes to be fully opaque and assign a fixed density value σ to each sample. Thanks to this representation, a single point per ray is sufficient to render the object.

3.1.4 *Deformation Modeling.*

Since the radiance field C alone supports only rendering of rigid objects expressed in a canonical space, to render articulated objects such as humans, we introduce a deformation model $\mathcal{D}$. Given an articulated object, we assume its kinematic tree is known and that the transformation $[\mathbf{R}_j | \mathbf{tr}_j]$ from each joint $j \in 1, \ldots, J$ to the parent joint is part of the object's properties. We then implement a deformation procedure based on **linear blend skinning (LBS)** [Lewis et al. 2000] and inspired by HumanNeRF [Weng et al. 2022], which employs the joint transformations and a learned volume of blending weights $\mathbf{W} \in \mathbb{R}^{J+1 \times H_W \times W_W \times D_W}$ to associate each point in the bounding box of the articulated object to the corresponding one in the canonical volume. We present additional details in Supplement C.

3.1.5 *Enhancer.*

NeRF models are often parametrized to output radiance $\mathbf{c} \in \mathbb{R}^3$ and directly produce an image. However, we find that such approach struggles to produce correct shading of the objects, with details such as shadows being difficult to synthesize. Also, to improve the computational efficiency of the method, we sample a limited number of points per ray that may introduce subtle artifacts in the geometry. To address these issues, we parametrize the model C to output features where the first three channels represent radiance and the subsequent represent learnable features. Then, we produce a feature grid $\mathbf{G} \in \mathbb{R}^{F \times H \times W}$ and an RGB image $\tilde{\mathbf{I}} \in \mathbb{R}^{3 \times H \times W}$. We introduce an enhancer network $\mathcal{F}$ modeled as a UNet [Ronneberger et al. 2015] architecture interleaved with weight demodulation layers [Karras et al. 2020] that maps $\mathbf{G}$ and the style codes $\boldsymbol{\omega}$ to the final RGB output $\hat{\mathbf{I}} \in \mathbb{R}^{3 \times H \times W}$.

3.1.6 *Object-specific Rendering.*

Our compositional approach allows the use of object-specific techniques. In particular, in the case of tennis, we detail in Supplement B how we can apply dedicated procedures to enhance the rendering quality of the ball, the racket, and the 2D user interfaces such as the scoreboards. The rendering of the tennis ball is treated specially to render the blur that occurs in real videos in the case of fast-moving objects. The racket can be inserted in a post-processing stage to compensate for the difficulty of NeRFs to render thin, fast-moving objects. Finally, the UI elements are removed from the scene, since they do not behave in a 3D-consistent manner. For Minecraft, we describe how the scene skybox is modeled.

3.1.7 Training. We train our model using reconstruction as the main driving signal. Given a frame $\mathbf{I}$ and reconstructed frame $\hat{\mathbf{I}}$, we use a combination of L2 reconstruction loss and the perceptual loss of Johnson et al. [2016] as our training loss. To minimize the alterations introduced by the enhancer and improve view consistency, we impose the same losses between $\mathbf{I}$ and $\tilde{\mathbf{I}}$, the output of the model before the feature enhancer. All losses are summed without weighting to produce the final loss term. To minimize GPU memory consumption, instead of rendering full images, we impose the losses on sampled image patches instead [Menapace et al. 2022].

We train all the components of the synthesis model jointly using Adam [Kingma and Ba 2015] for 300k steps with batch size 32. We set the learning rate to $1e-4$ and exponentially decrease it to $1e-5$ at the end of training. The framework is trained on videos with 1024×576 px resolution. We present additional details in Supplement D.1 and in Supplement E.1 and discuss inference details in Supplement F.

3.2 Animation Model

In this section, we describe the animation model (see Figure 2(b)), whose task is that of generating sequences of states $\mathbf{s} \in \mathbb{S}^T$ according to user inputs. The animation model allows users to specify conditioning signals, or prompts, in two forms. First, conditional signals can take the form of values that the user wants to impose on some object properties in the sequence, such as the player position at a certain timestep. This signal is represented by a sequence $\mathbf{s}^c \in \mathbb{S}^T$. This form of conditioning allows fine control over the sequence to generate but requires directly specifying values of properties. Second, to allow high-level, yet granular control over the sequence, we introduce actions in the form of text $\mathbf{a}^c \in \mathbb{L}^{A\times T}$ that specify the behavior of each of the A actionable objects at each timestep in the sequence, where $\mathbb{L}$ is the set of all strings of text. To maximize the flexibility of the framework, we consider all values in $\mathbf{s}^c$ and $\mathbf{a}^c$ to be optional, thus we introduce their respective masks $\mathbf{m}^{\mathbf{s}} \in \{0,1\}^{P\times T}$ and $\mathbf{m}^{\mathbf{a}} \in \{0,1\}^{A\times T}$ that are set to 1 when the respective conditioning signal is present. We assume elements where the mask is not set to be equal to 0. The animation model predicts $\mathbf{s}^p \in \mathbb{S}^T$ conditioned on $\mathbf{s}^c$ and $\mathbf{a}^c$ such that:

$$\mathbf{s} = \mathbf{s}^p + \mathbf{s}^c, \tag{1}$$

where we consider the entries in $\mathbf{s}^p$ and $\mathbf{s}^c$ to be mutually exclusive, i.e., an element of $\mathbf{s}^p$ is 0 if the corresponding conditioning signal in $\mathbf{s}^c$ is present according to $\mathbf{m}^{\mathbf{s}}$. Note that the prediction of actions is not necessary, since $\mathbf{s}$ is sufficient for rendering.

We adopt a temporal model $\mathcal{A}$ based on a non-autoregressive masked transformer design and leverage the knowledge of a pretrained language model in a text encoder $\mathcal{T}$ to model action conditioning information [Han et al. 2022]. The masked design provides support for the optional conditioning signals and is trained using masked sequence modeling, where we sample $\mathbf{m}^{\mathbf{s}}$ and $\mathbf{m}^{\mathbf{a}}$ according to various strategies that emulate desired inference tasks.

In Section 3.2.1, we define our text encoder, Section 3.2.2 defines the diffusion backbone, and in Section 3.2.3, we describe the training procedure.

3.2.1 Text Encoder. We introduce a text encoder $\mathcal{T}$ that encodes textual actions into a sequence of fixed-size text embeddings:

$$\mathbf{a}^{\text{emb}} = \mathcal{T}(\mathbf{a}^c) \in \mathbb{R}^{A\times T\times N_t}, \tag{2}$$

where N_t is the size of the embedding for the individual sentence. Given a textual action, we leverage a pretrained T5 text model [Raffel et al. 2022] $\mathcal{T}_{\text{enc}}$ that tokenizes the sequence and produces an output feature for each token. Successively, a feature aggregator $\mathcal{T}_{\text{agg}}$ modeled as a transformer encoder [Vaswani et al. 2017] produces the aggregated text embedding from the text model features. To retain existing knowledge into $\mathcal{T}_{\text{enc}}$, we keep it frozen and only train the feature aggregator $\mathcal{T}_{\text{agg}}$.

3.2.2 Temporal Modeling. In this section, we introduce the temporal model $\mathcal{A}$ that predicts the sequence of states $\mathbf{s}$ conditioned on known state values $\mathbf{s}^c$, action embeddings $\mathbf{a}^{\text{emb}}$, and the respective masks $\mathbf{m}^{\mathbf{s}}$ and $\mathbf{m}^{\mathbf{a}}$. Since only unknown state values need to be predicted, the model predicts $\mathbf{s}^p$ and the complete sequence of states is obtained as $\mathbf{s} = \mathbf{s}^p + \mathbf{s}^c$, following Equation (1). Diffusion models have recently shown state-of-the-art performance on several tasks closely related to our setting such as sequence modeling [Tashiro et al. 2021] and text-conditioned human motion generation [Dabral et al. 2023; Zhang et al. 2022]. Thus, we follow the DDPM [Ho et al. 2020] diffusion framework, and we frame the prediction of $\mathbf{s}^p = \mathbf{s}^p_0$ as a progressive denoising process $\mathbf{s}^p_0, \ldots, \mathbf{s}^p_K$, where we introduce the diffusion timestep index $k \in 0, \ldots, K$. The temporal model $\mathcal{A}$ acts as a noise estimator that predicts the Gaussian noise $\boldsymbol{\epsilon}_k$ in the noisy sequence of unknown states $\mathbf{s}^p_k$ at diffusion timestep k:

$$\boldsymbol{\epsilon}^p_k = \mathcal{A}\left(\mathbf{s}^p_k | \mathbf{s}^c, \mathbf{a}^{\text{emb}}, \mathbf{m}^{\mathbf{s}}, \mathbf{m}^{\mathbf{a}}, k\right). \tag{3}$$

An illustration of the proposed diffusion model is shown in Figure 2(b).

We realize $\mathcal{A}$ using a transformer encoder [Vaswani et al. 2017]. To prepare the transformer's input sequence, we employ linear projection layers $\mathcal{P}$ with separate parameters for each object property. Since corresponding entries in $\mathbf{s}^p_k$ and $\mathbf{s}^c$ are mutually exclusive, we only consider the one that is present as input to the transformer, and we employ different projection parameters to enable the model to easily distinguish between the two. An analogous projection is performed for $\mathbf{a}^{\text{emb}}$ and, subsequently, the projection outputs for states and actions are concatenated into a single sequence $\mathbf{e} \in \mathbb{R}^{P+A\times T\times E}$, which constitutes the input to the transformer. An output projection layer with separate weights for each object property produces the prediction $\boldsymbol{\epsilon}^p_k$ at the original dimensionality. To condition the model on the diffusion timestep k, we introduce a weight demodulation layer [Karras et al. 2020] after each self-attention and feedforward block [Zhang et al. 2022].

To model long sequences while keeping reasonable computational complexity and preserving the ability to model long-term relationships between sequence elements, it is desirable to build the sequences using states sampled at a low framerate. However, this strategy would not allow the model to generate content at the original framerate and would prevent it from understanding dynamics such as limb movements that are clear only when observing sequences sampled at high framerates. To address this issue, we use the weight demodulation layers to further condition our model on the sampling framerate ν to enable a progressive increase of the framerate at inference time (see Supplement F.2.1).

3.2.3 Training. To train our model, we sample a sequence $\mathbf{s}$ with corresponding actions $\mathbf{a}$ from a video in the dataset at a uniformly sampled framerate ν. Successively, we obtain masks $\mathbf{m}^{\mathbf{s}}$ and $\mathbf{m}^{\mathbf{a}}$ according to masking strategies we detail in Supplement E.2. The sequences for training are obtained following $\mathbf{s}_0^p = \mathbf{s} \odot (1 - \mathbf{m}^{\mathbf{s}})$ and $\mathbf{s}^c = \mathbf{s} \odot \mathbf{m}^{\mathbf{s}}$, and actions as $\mathbf{a}^c = \mathbf{a} \odot \mathbf{m}^{\mathbf{a}}$, where $\odot$ denotes the Hadamard product.

We train our model by minimizing the DDPM [Ho et al. 2020] training objective:

$$\mathbb{E}_{k\sim\mathcal{U}(1,K),\epsilon\sim\mathcal{N}(0,I)} ||\boldsymbol{\epsilon}_k^p - \boldsymbol{\epsilon}_k||, \tag{4}$$

where $\boldsymbol{\epsilon}_k^p$ is the noise estimated by the temporal model $\mathcal{A}$ according to Equation (3). Note that the loss is not applied to positions in the sequence corresponding to conditioning signals [Tashiro et al. 2021].

Our model is trained using the Adam [Kingma and Ba 2015] optimizer with a learning rate of $1e-4$, cosine schedule, and with 10k warmup steps. We train the model for a total of 2.5M steps and a batch size of 32. We set the length of the training sequences to $T = 16$. The number of diffusion timesteps is set to $K = 1{,}000$ and we adopt a linear noise schedule [Ho et al. 2020]. Additional details are presented in Supplement D.2 and Supplement F.

4 APPLICATIONS

Our framework enables a series of applications that are unlocked by its expressive state representation, the possibility to render it using a 3D-aware synthesis model, and the ability to generate sequences of states with an animation model that understands the game dynamics and can be conditioned on a wide range of signals. In the following, we demonstrate a set of selected applications.

Our state representation is modular, where the style is one of the components. Style swapping is enabled by swapping the style of the desired object $\boldsymbol{\omega}$ in the original image with the one from a target image. Similarly to a traditional game engine, our synthesis model renders the current state of the environment from a user-defined perspective. This enables our model to perform novel view synthesis. We show in Supplement G examples of both these capabilities.

We now show a set of applications enabled by the animation model. In Figure 3, we show results for generating different sequences using textual actions starting from a common initial state. Thanks to the textual action representation, it is possible to gain fine control over the generated results and to make use of referential language.

Our animation model, however, is not limited to generate sequences given step-by-step actions. Thanks to its understanding of the game's dynamics, the model can tackle more complex tasks such as modeling an opponent against which a user-controlled player can play (see Figure 4), or even controlling all players without user intervention (see Figure 5) in a way similar to a "game AI."

The animation model also unlocks the "director's mode," where the user can generate sequences by specifying prompts consisting in a desired set of high-level constraints or goals. The model is able to reason on actions to find a solution satisfying the given constraints. As a first example, Figure 6 demonstrates results for a navigation problem, where the user specifies a desired initial and final player position in the scene, and the model devises a path between them. Notably, the user can also constrain the solution on intermediate waypoints by means of natural language. As a second example, Figure 7 shows that the model is capable of devising strategies to defeat an opponent. Given an original sequence where the player commits a mistake and loses, the model can devise which actions the player should have taken to win. Notably, these model capabilities are learned by just observing sequences annotated with textual actions.

5 EVALUATION

In this section, we introduce our Tennis and Minecraft datasets (Section 5.1), describe our experimental protocol (Section 5.2), and perform evaluation of both the synthesis model (Section 5.3) and the animation model (Section 5.4). Additional evaluation results are shown in Supplement H.

5.1 Datasets

We collect two datasets to evaluate our method. Both datasets and the employed data collection tools are publicly available. In the following, we describe their structure and the available annotations.

5.1.1 Tennis Dataset. We collect a dataset of broadcast tennis matches starting from the videos in Menapace et al. [2022]. The dataset depicts matches between two professional players from major tennis tournaments, captured with a single, static bird's eye camera.

To enable the construction of PGMs, we collect a wide range of annotations with a combination of manual and automatic methods (see Supplement A.1):

— For each frame, we perform camera calibration.
— For each of the two players, we perform tracking and collect full SMPL [Loper et al. 2015] body parameters. Note that, in our work, we only use a subset of the parameters: rotation and translation associated with each joint and the location of the root joint in the scene.
— For each player and frame, we manually annotate textual descriptions of the action being performed. We structure captions so each includes information on where and how the player is moving, the particular type of tennis shot being performed, and the location where the shot is aimed (see Supplement A.4). Captions make use of technical terms to describe shot types and field locations. In contrast to other video-text datasets that contain a single video-level [Bain et al. 2021] or high-level action descriptions weakly aligned with video content [Miech et al. 2019], the captions in our dataset are separate for each object and constitute a fine-grained description of the actions taking place in the frame.
— For the ball, we perform 3D tracking and provide its position in the scene and its velocity vector indicating the speed and direction of movement.

We collect 7,112 video sequences in 1920 × 1080 px resolution and 25 fps starting from the videos in Menapace et al. [2022] for a total duration of 15.5 h. The dataset features 1.12M fully

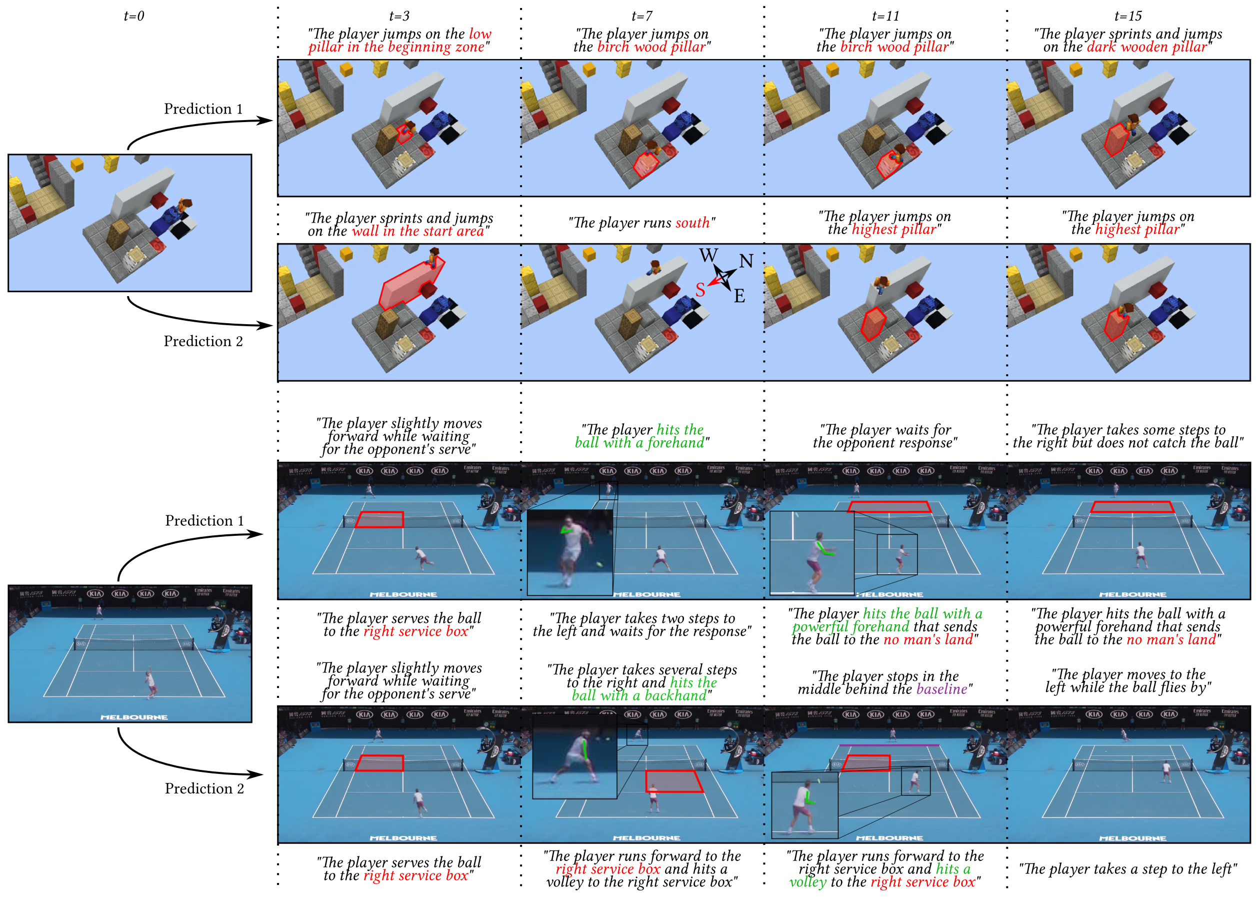

t=0
t=3
"The player jumps on the low pillar in the beginning zone"
t=7
"The player jumps on the birch wood pillar"
t=11
"The player jumps on the birch wood pillar"
t=15
"The player sprints and jumps on the dark wooden pillar"
Prediction 1
"The player sprints and jumps on the wall in the start area"
"The player runs south"
W
N
S
E
"The player jumps on the highest pillar"
"The player jumps on the highest pillar"
Prediction 2
"The player slightly moves forward while waiting for the opponent's serve"
"The player hits the ball with a forehand"
"The player waits for the opponent response"
"The player takes some steps to the right but does not catch the ball"
Prediction 1
"The player serves the ball to the right service box"
"The player takes two steps to the left and waits for the response"
"The player hits the ball with a powerful forehand that sends the ball to the no man's land"
"The player hits the ball with a powerful forehand that sends the ball to the no man's land"
"The player slightly moves forward while waiting for the opponent's serve"
"The player takes several steps to the right and hits the ball with a backhand"
"The player stops in the middle behind the baseline"
"The player moves to the left while the ball flies by"
Prediction 2
"The player serves the ball to the right service box"
"The player runs forward to the right service box and hits a volley to the right service box"
"The player runs forward to the right service box and hits a volley to the right service box"
"The player takes a step to the left"

Fig. 3. Different sequences predicted on the Tennis and Minecraft datasets starting from the same initial state and altering the text conditioning. Our model moves players and designates shot targets using domain-specific referential language (e.g., "*right service box*," "*no man's land*," "*baseline*"). The model supports fine-grained control over the various tennis shots using technical terms (e.g., "*forehand*," "*backhand*," "*volley").*

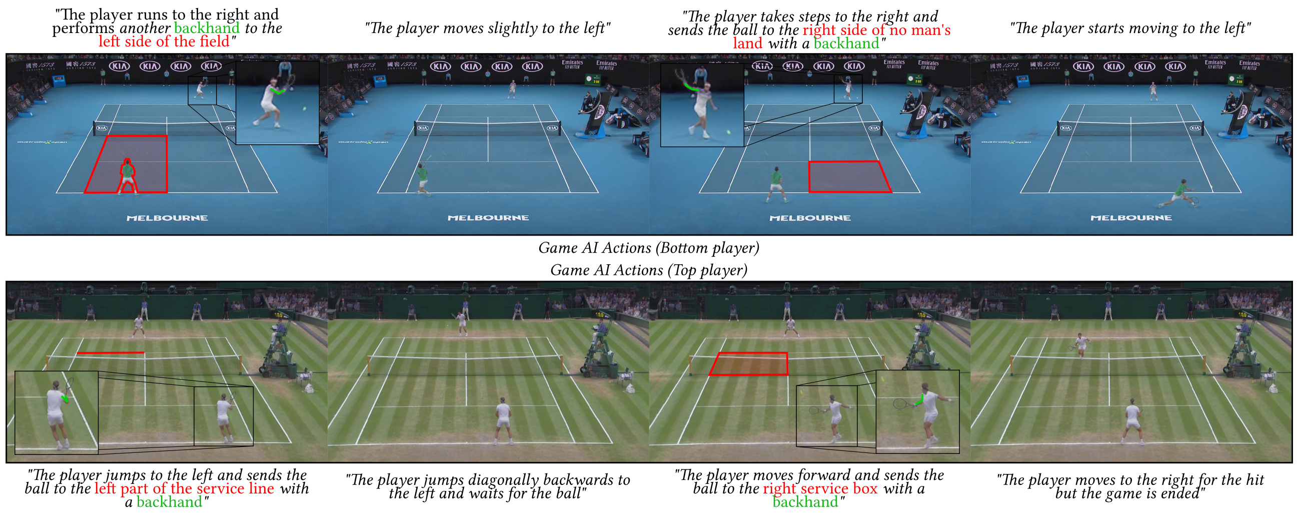

"The player runs to the right and performs another backhand to the left side of the field"
"The player moves slightly to the left"
"The player takes steps to the right and sends the ball to the right side of no man's land with a backhand"
"The player starts moving to the left"
Game AI Actions (Bottom player)
Game AI Actions (Top player)
"The player jumps to the left and sends the ball to the left part of the service line with a backhand"
"The player jumps diagonally backwards to the left and waits for the ball"
"The player moves forward and sends the ball to the right service box with a backhand"
"The player moves to the right for the hit but the game is ended"

Fig. 4. Sequences generated by specifying actions for one of the players and letting the model act as the game AI and take control of the opponent. The game AI successfully responds to the actions of the player by running to the right (see top sequence) or towards the net (see bottom sequence), following two challenging shots of the user-controlled player.

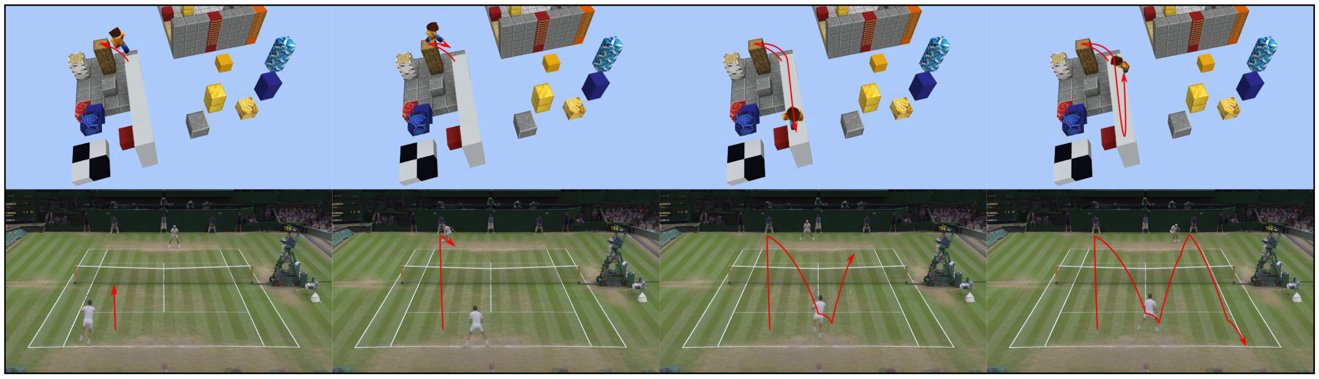


Fig. 5. Sequences generated without any user conditioning signal. The actions of all players are controlled by the model that acts as the game AI. In tennis, the players produce a realistic exchange, with the bottom player advancing aggressively toward the net and the top player defeating him with a shot along the right sideline. The Minecraft player and tennis ball trajectories are highlighted for better visualization.

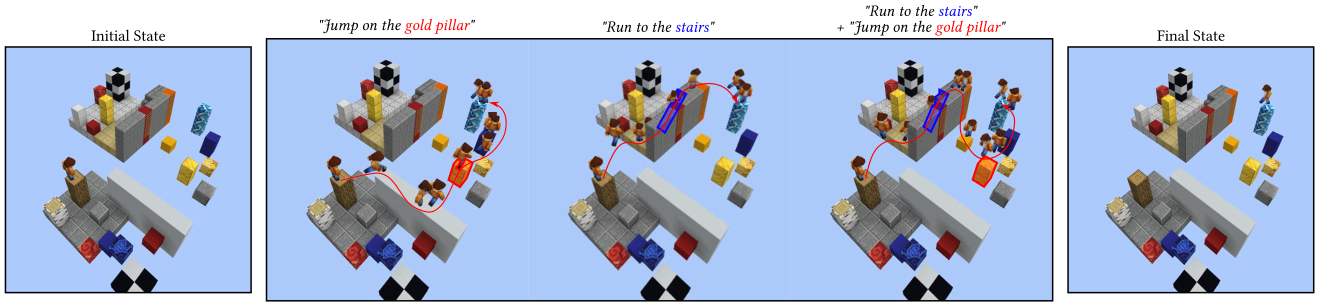


Fig. 6. Given an initial and a final state, we generate all the states in between. We repeat the generation multiple times, conditioning it using different actions indicating the desired intermediate waypoints.

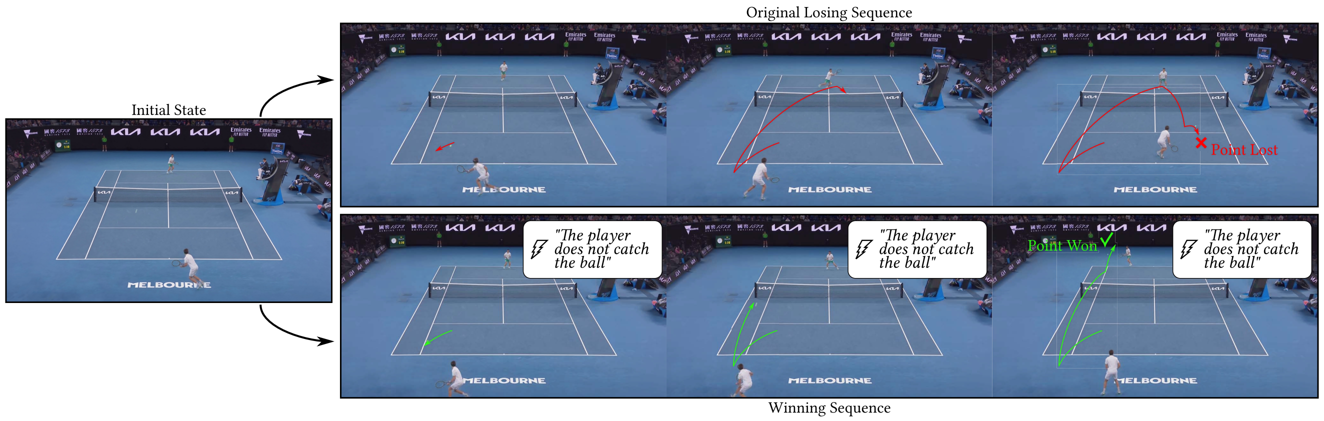

Original Losing Sequence
Initial State
Point Lost
"The player does not catch the ball"
"The player does not catch the ball"
Point Won
"The player does not catch the ball"
Winning Sequence

Fig. 7. Given a sequence where the bottom player loses (see top), we ask the model to modify it such that the bottom player wins instead (see bottom). To do so, we condition the top player on the action "*The player does not catch the ball.*" While in the original sequence the bottom player aims its response to the center of the field where the opponent is waiting, the model now successfully generates a winning set of moves for the bottom player that sends the ball along the left sideline, too far for the top player to reach.

annotated frames and 25.5k unique captions with 915 unique words. We highlight key statistics of the dataset and show samples in Supplement A.

We note that broadcast Tennis videos are monocular and do not feature camera movements other than rotation, thus the dataset does not make it possible to recover the 3D geometry of static objects [Menapace et al. 2022].

5.1.2 Minecraft Dataset. We collect a synthetic dataset from the Minecraft video game. This dataset depicts a player performing

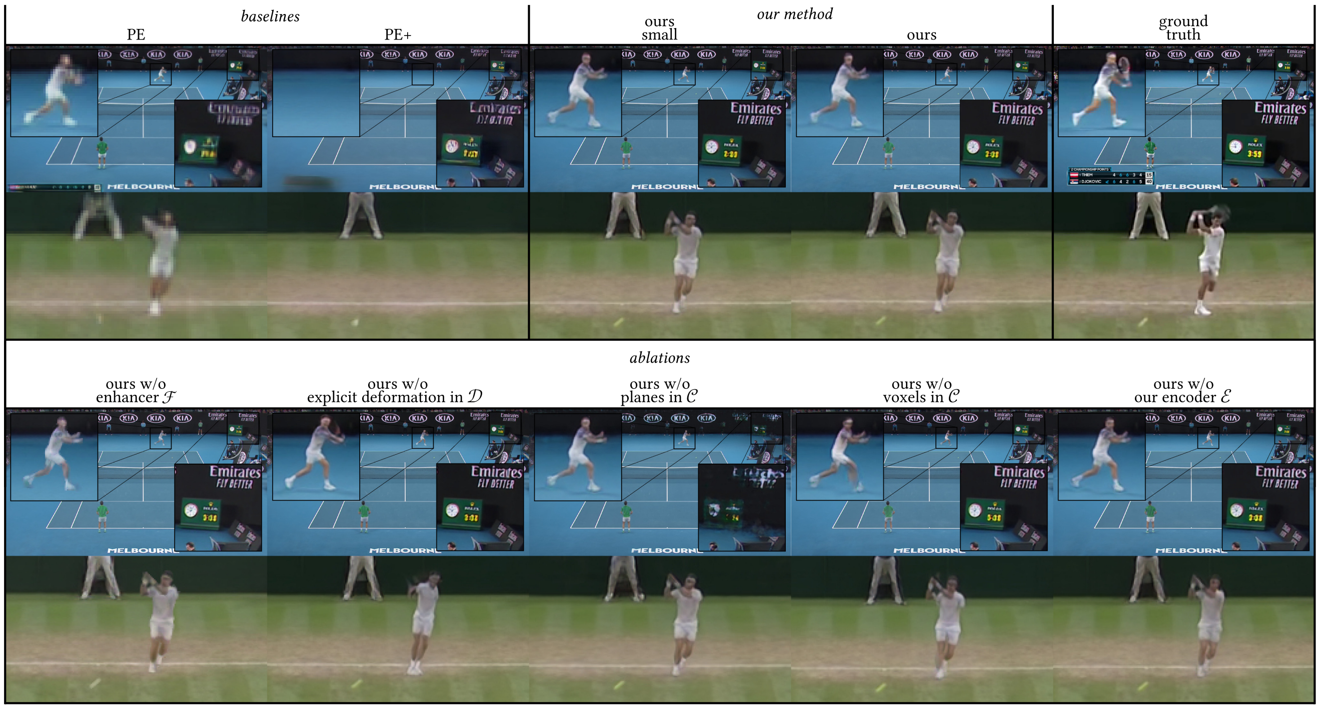

baselines
PE
PE+
our method
ours small
ours
ground truth
ablations
ours w/o enhancer $\mathcal{F}$
ours w/o explicit deformation in $\mathcal{D}$
ours w/o planes in $\mathcal{C}$
ours w/o voxels in $\mathcal{C}$
ours w/o our encoder $\mathcal{E}$

Fig. 8. Synthesis model qualitative results on the Tennis dataset. Compared to PE [Menapace et al. 2022], our model generates sharper players and static scene elements. Our ablation study shows corruption of the player geometry when voxels or our deformation model are not used. When removing our canonical plane representation, static scene elements appear blurry. When our feature enhancer is removed, the model does not generate shadows and players lose quality.

a series of complex movements in a static Minecraft world that includes walking, sprinting, jumping, and climbing on various world structures such as platforms, pillars, stairs, and ladders. A single, monocular camera that slowly orbits around the scene center is used to capture the scenes. We collect a range of synthetic annotations using a game add-on we develop starting from ReplayMod [2022]:

- Camera calibration for each frame.
- Player rotation and translation parameters associated with each joint in the Minecraft kinematic tree format and the location of the root joint in the scene (see Supplement A.2).
- A synthetically generated text caption describing the action being performed by the player. We assign varied, descriptive names to each element of the scene and build captions that describe scene elements or directions towards which the player is moving. Additionally, our captions capture how movement is happening, i.e., by jumping, sprinting, walking, climbing, or falling. We adopt a stochastic caption generation procedure that generates multiple alternative captions for each frame.

A total of 61 videos are collected in 1024 × 576 px resolution and 20 fps for a total duration of 1.21 h. The dataset contains 68.5k fully annotated frames and 1.24k unique captions with 117 unique words. We highlight key statistics for the dataset in Supplement A.

5.2 Evaluation Protocol

We evaluate the synthesis and the animation models separately, following a similar evaluation protocol. We divide the test dataset into non-overlapping sequences of 16 frames sampled at 5 fps and 4 fps, respectively, for the Minecraft and Tennis datasets and make use of the synthesis or animation model to reconstruct them. In the case of the synthesis model, we directly reconstruct the video frames and compute the following metrics:

- *LPIPS* [Zhang et al. 2018] is a standard metric for evaluating the reconstruction quality of the generated images.
- *FID* [Heusel et al. 2017] is a widely used metric for image generation quality.
- *FVD* [Unterthiner et al. 2018] is a standard metric for assessing the quality of generated videos.
- ***Average Detection Distance (ADD)*** [Menapace et al. 2021] measures the average distance in pixels between the bounding box centers of ground truth bounding boxes and bounding boxes obtained from the generated sequences through a pretrained detector [Ren et al. 2015].
- ***Missing Detection Rate (MDR)*** [Menapace et al. 2021] estimates the rate of bounding boxes that are present in the ground truth but that are missing in the generated videos

For the animation model, we evaluate reconstruction of the object properties. Note that different strategies for masking affect the behavior of the model and the nature of the reconstruction task, thus we separately evaluate different masking configurations corresponding to different inference tasks. We compute metrics that address both the fidelity of the reconstruction and the realism of the produced sequences:

— *L2* computes the fidelity of the reconstruction by measuring the distance between the ground truth and reconstructed object properties along the sequence.
— ***Fréchet Distance (FD)*** [Fréchet 1957] measures the realism of each object property by computing the Fréchet Distance between the distribution of real sequences of a certain object property and of generated ones.

We select different reconstruction tasks for evaluation:

— *Video prediction conditioned on actions* consists in reconstructing the complete sequence starting from the initial state while the actions are specified for all timesteps. This setting corresponds to the evaluation setting of Menapace et al. [2022].
— *Unconditioned video prediction* consists in reconstructing the complete sequence starting from the first state only.
— *Opponent modeling* consists in reconstructing the object properties of an unknown player, based on the state of the other player, with actions specified only on the known player. Good performance in this task indicates the ability to model an opponent against which a user can play.
— *Sequence completion* consists in reconstructing a sequence where eight consecutive states are missing. No actions are specified for the missing states. Good performance in this task indicates ability in reasoning on how it is possible to reach a certain goal state starting from the current one.

5.3 Synthesis Model Evaluation

In this section, we evaluate the performance of the synthesis model.

5.3.1 Comparison to Baselines. We evaluate our method against **Playable Environments (PE)** [Menapace et al. 2022], the work most related to ours in that it builds a controllable 3D environment representation that is rendered with a compositional NeRF model where the position of each object is given and pose parameters are treated as a latent variable. Since the original method supports only outputs at 512×288 px resolution, we produce baselines trained at both 512×288 px and 1024×576 px resolution, which we name PE and PE+, respectively. For a fair comparison, we also introduce in the baselines our same mechanism for representing ball blur and train a variant of our model using the same amount of computational resources as the baselines (Ours Small).

Results of the comparison are shown in Table 1, while qualitative results are shown in Figure 8. Our method scores best in terms of LPIPS, ADD, and MDR. Compared to PE+, our method produces significantly better FID and FVD scores. As shown in Figure 8, PE and PE+ produce checkerboard artifacts that are particularly noticeable on static scene elements such as judge stands, while our method produces sharp details. We attribute this difference to our ray sampling scheme and feature enhancer design that, in contrast to PE, do not sample rays at low resolution and perform upsampling, but rather directly operate on high resolution. In addition, thanks to our deformation and canonical space modeling strategies and higher resolution, our method produces more detailed players with respect to PE, where they frequently appear with missing limbs and blurred clothing. Finally, our model produces a realistic ball, while PE struggles to correctly model small objects, presumably due to its upsampling strategy that causes rays to be sampled more sparsely and thus do not intersect with the ball frequently enough to correctly render its blur effect.

Table 1. Comparison with Baselines and Ablation of the Synthesis Model

Tennis	LPIPS↓	FID↓	FVD↓	ADD↓	MDR↓
PE[†] [Menapace et al. 2022]	0.188	11.5	349	3.74	0.200
PE+ [Menapace et al. 2022]	0.232	40.4	2432	132.3	49.7
w/o enhancer $\mathcal{F}$	0.167	15.6	570	3.02	0.0728
w/o explicit deformation in $\mathcal{D}$	0.156	13.3	524	3.10	0.0587
w/o planes in $\mathcal{C}$	0.241	30.4	1064	2.94	0.0611
w/o voxels in $\mathcal{C}$	0.170	17.1	757	3.03	0.0399
w/o our encoder $\mathcal{E}$	0.174	15.0	600	3.18	0.0564
Ours Small	0.156	13.4	523	2.88	0.0470
Ours	0.152	12.8	516	2.88	0.0423
Minecraft	LPIPS↓	FID↓	FVD↓	ADD↓	MDR↓
PE[†] [Menapace et al. 2022]	0.0235	13.9	21.5	5.77	0.0412
PE+ [Menapace et al. 2022]	0.0238	15.5	51.7	120.6	0.939
Ours Small	0.00996	3.56	8.83	2.02	0.0529
Ours	0.00814	2.81	7.08	1.98	0.0508

MDR in %, ADD in pixels. Note that FID and FVD are computed on images downscaled to the feature extractor training resolution, thus blurriness in the PE baseline caused by its reduced resolution is not captured by these metrics. LPIPS correctly reflects lack of sharpness in the PE results (see Figure 8). †denotes output in 512×288 px rather than 1024×576 px resolution.

5.3.2 Ablation. To validate our design choices, we produce several variations of our method, each produced by removing one of our proposed architectural elements: We remove the enhancer $\mathcal{F}$ and directly consider $\tilde{\mathbf{I}}$ as our output; we remove the explicit deformation modeling procedure in $\mathcal{D}$ of Section 3.1.4 and substitute it with an MLP directly predicting the deformation using a learnable pose code as in Menapace et al. [2022]; Tretschk et al. [2021]; we remove the plane-based canonical volume representation in $\mathcal{C}$ for planar objects and use an MLP instead; we remove the voxel-based volume representation in $\mathcal{C}$ and use an MLP instead; we substitute our style encoder $\mathcal{E}$ with an ad hoc encoder for each object in the scene, following Menapace et al. [2022].

We perform the ablation on the Tennis dataset and show results in Table 1 and Figure 8. To reduce computation, we train the ablation models using the same hyperparameters as the "Ours Small" model.

When removing the enhancer $\mathcal{F}$, our model produces players with fewer details and does not generate shadow effects below players (see first row in Figure 8). When our deformation modeling procedure is not employed, the method produces comparable LPIPS, FID, and FVD scores, but an analysis of the qualitatives shows that players may appear with corrupted limbs (see last row in Figure 8). In addition, the use of such learned pose representation would reduce the controllability of the synthesis model with respect to the use of an explicit kinematic tree. When plane-based or voxel-based canonical modeling is removed, we notice artifacts in the static scene elements, such as corrupted logos, and in the players, such as detached or doubled limbs. Finally, when we replace our style encoder design with the one of Menapace et al. [2022], we notice fewer details in scene elements.

5.4 Animation Model Evaluation

In this section, we evaluate the performance of the animation model.

5.4.1 Comparison to Baselines. Similarly to the synthesis model, we compare our animation model against the one of **Playable Environments (PE)** [Menapace et al. 2022], the most related to our work, since it operates on a similar environment representation. While the baseline jointly learns discrete actions and generates sequences conditioned on such actions, we assume the text action representations to be available in our task, so, for fairness of evaluation, we introduce our same text encoder $\mathcal{T}$ in the baseline to make use of the action information. To reduce computation, we perform the comparison using half of the computational resources and a reduced training schedule; consequently, we also retrain our model, producing a reduced variant (Ours Small). To render results, we always make use of our synthesis model.

We show results averaged over all inference tasks in Table 2 and report the results for each task in Supplement H.2. Our method outperforms the baseline in all evaluation tasks according to both L2 and FD metrics. From the qualitative results in Figure 9 and in accordance with the FD metrics, we notice that our method produces more realistic player poses with respect to PE that tends to keep player poses close to the average pose and to slide the players on the scene. We attribute this difference to the use of the diffusion framework in our method. Consider the example of generating a player walking forward. It is equally probable that the player moves the left or right leg first. In the case of a reconstruction-based training objective such as the main one of PE, the model is encouraged to produce an average leg movement result that consists in not moving the legs at all. However, diffusion models learn the multimodal distributions of the motion, thus they are able to sample one of the possible motions without averaging its predictions.

5.4.2 Ablation. To validate this hypothesis and demonstrate the benefits of our diffusion formulation, we produce two variations of our method. The first substitutes the diffusion framework with a reconstruction objective, keeping the transformer-based architecture unaltered. The second, in addition to using the reconstruction objective, models $\mathcal{A}$ using an LSTM, similarly to the PE baseline. Differently from the PE baseline, however, this variant does not make use of adversarial training and employs a single LSTM model for all objects, rather than a separate model for each.

We show results in Table 2. Our model consistently outperforms the baselines in terms of FD, showing a better ability to capture realistic sequences. Consistently with our assessment in Section 5.4.1, Figure 9 shows that our method trained with a reconstruction objective produces player movement with noticeable artifacts analogously to PE, validating the choice of the diffusion framework.

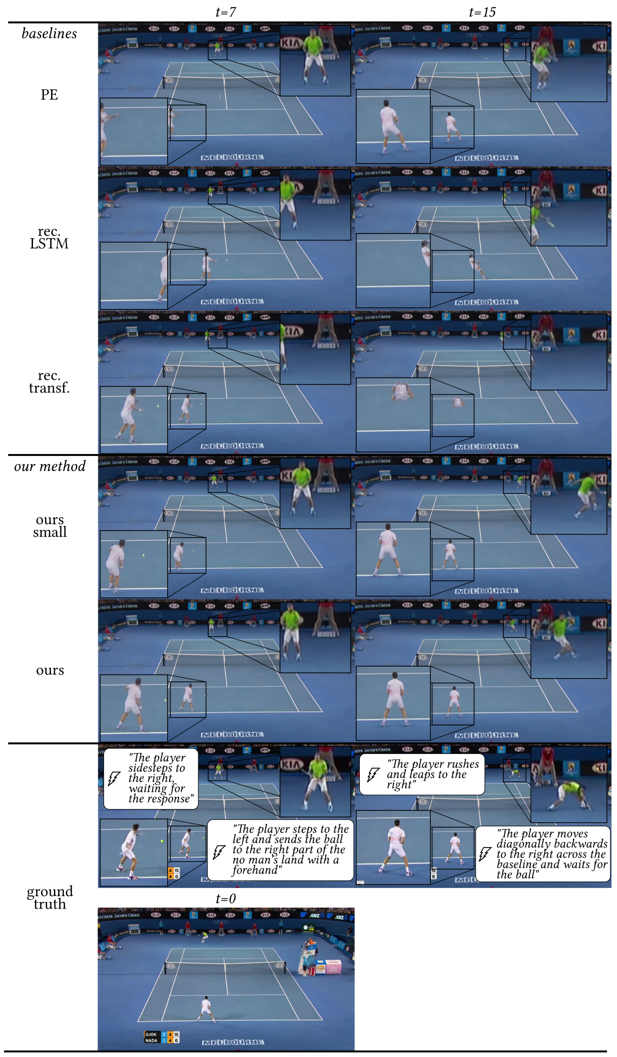

t=7
t=15
baselines
PE
rec. LSTM
rec. transf.
our method
ours small
ours
ground truth
"The player sidesteps to the right, waiting for the response"
"The player rushes and leaps to the right"
"The player steps to the left and sends the ball to the right part of the no man's land with a forehand"
"The player moves diagonally backwards to the right across the baseline and waits for the ball"
t=0

Fig. 9. Qualitative results on the Tennis dataset. Sequences are produced in a video prediction setting that uses the first frame object properties and all actions as conditioning. The location of players is consistently closer to the ground truth for our method. Our method captures the multimodal distribution of player poses and generates vivid limb movements, while the baselines produce poses as the average of the distribution, resulting in reduced limb movement and tilted root joints. Additional samples are shown in Supplement H.2.

5.5 Limitations

Since the model is trained on a dataset showing only plausible actions, the model's behavior is not defined when an *implausible* action is specified, such as hitting a ball while moving in the wrong direction to intercept it or jumping on a pillar that is out of reach. In these cases, we find the model to ignore the implausible part of the command and produce the closest plausible command or, less frequently, to produce implausible outcomes such as irrealistic long jumps (see Figure 10). In addition, the model does not generate actions extremely out of distribution such as performing a backflip or doing a push-up. This aspect could be addressed by jointly training the animation model on multiple diverse datasets, which we consider an interesting future direction.

While our Tennis dataset contains varied text annotations that allow the model to generalize to text inputs with varied structure, our Minecraft dataset's synthetic text annotations are less varied, and the fixed synthetic structure of sentences tends to be memorized, making the model less effective if a different syntax is used

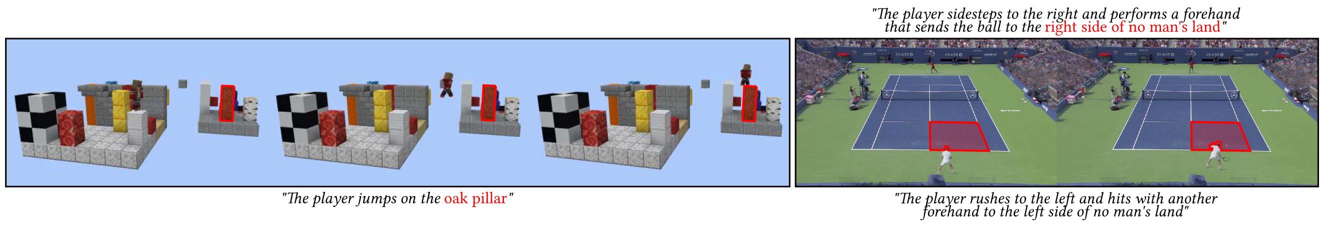

"The player sidesteps to the right and performs a forehand that sends the ball to the right side of no man's land"
"The player jumps on the oak pillar"
"The player rushes to the left and hits with another forehand to the left side of no man's land"

Fig. 10. Behavior of the model when implausible actions are provided. In the left example, the model generates an irrealistic long jump to reach the specified pillar. In the right example, the bottom player is instructed to move left to intercept a ball coming to his right. In this case, the left movement command is ignored by the model to produce the closest plausible outcome.

Table 2. Animation Model Comparison with Baselines and Ablation with Results Averaged over All Inference Tasks

Tennis	*Position*		*Root angle*		*Joints 3D*	
	L2↓	FD↓	L2↓	FD↓	L2↓	FD↓
PE	3.291	229.112	1.126	15.953	0.303	53.242
Rec. LSTM	1.597	7.253	0.907	7.051	0.193	16.735
Rec. Transf.	1.074	4.402	0.767	6.838	0.175	14.845
Ours Small	1.380	1.443	1.014	0.560	0.148	1.253
Ours	1.099	0.929	0.844	0.356	0.129	0.836
Minecraft	*Position*		*Root angle*		*Joints 3D*	
	L2↓	FD↓	L2↓	FD↓	L2↓	FD↓
PE	2.739	105.973	1.620	31.232	0.311	39.572
Rec. LSTM	2.292	47.296	1.702	49.971	0.489	99.843
Rec. Transf.	2.154	53.198	1.430	36.123	0.385	69.977
Ours Small	1.084	4.461	1.077	6.016	0.140	3.590
Ours	1.065	4.815	0.956	4.083	0.132	3.360

Position and Joints 3D in meters, Root angle in axis-angle representation.

(see Section H.1). To address this issue, a more sophisticated algorithm can be employed to generate action annotation on the Minecraft dataset.

Our model learns to associate referential language to scene coordinates rather than the appearance of the referred object, and the model memorizes the position of contact surfaces. While tennis scenes always have the same structure, for Minecraft the model cannot generalize to different scenes. This concern can be addressed by conditioning the animation model on the scene's geometry, which we leave as future work.

We find our animation model to overfit to the Tennis dataset when less than 60% of the training data is used (see Supplement H.4). We leave as an interesting avenue of future work the investigation of regularization techniques such as dropout or weight decay, which have the potential to reduce overfitting in this scenario.

Our animation model outperforms baselines that operate under the same data assumptions [Menapace et al. 2022] in terms of animation quality. With respect to recent character animation methods [Holden et al. 2020; Starke et al. 2019, 2020] making use of richly annotated motion capture data and dataset-specific handcrafted optimizations (see Section 2.3), our method demonstrates more advanced game dynamics and game AI modeling capabilities but produces foot-sliding artifacts. We expect continuous improvements in diffusion models to alleviate such artifacts and expect further improvements by considering different parametrizations of pose parameters taking into consideration the distance of limbs from the terrain, which we will explore in future work.

Last, our animation model does not yet produce results in real-time. We discuss inference speed and strategies to make the model real-time in Supplement F.1. Improving the sampling speed of diffusion models is an actively investigated problem [Meng et al. 2022; Salimans and Ho 2022; Song et al. 2021] that is orthogonal to ours.

6 CONCLUSIONS

In this article, we demonstrate the feasibility of learning game models able to answer challenging user prompts and show that textual action representations are critical for unlocking fine-grained control over the generation process and enabling compelling constraint- and goal-driven generation applications. These results, jointly with two richly annotated text-video datasets, pave the way towards learning game models for complex, real-world scenes.

ACKNOWLEDGEMENTS

We would like to thank Christian Theobalt for his feedback on the manuscript draft, Denys Poluyanov, Eugene Shevchuk, and Oleksandr Pyshchenko for the useful discussion and validation of the use cases of PGMs, Maryna Diakonova for her support in data labeling, and Anton Kuzmenko and Vadym Hrebennyk for their assistance in creating the accompanying video.

REFERENCES

Panos Achlioptas, Ian Huang, Minhyuk Sung, Sergey Tulyakov, and Leonidas Guibas. 2023. ChangeIt3D: Language-assisted 3D shape edits and deformations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'23).*

Nikos Athanasiou, Mathis Petrovich, Michael J. Black, and Gül Varol. 2022. TEACH: Temporal action compositions for 3D humans. In *Proceedings of the International Conference on 3D Vision (3DV'22).*

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. 2018. Stochastic variational video prediction. In *Proceedings of the International Conference on Learning Representations (ICLR'18).*

Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. 2021. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'21).*

Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. 2023. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'23).*

Michael Büttner and Simon Clavet. 2015. Motion matching—The road to next gen animation. In *Proceedings of Nucl.AI.*

Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. 2022. Efficient geometry-aware 3D generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'22).*

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. TensoRF: Tensorial radiance fields. In *Proceedings of the European Conference on Computer Vision (ECCV'22).*

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J. Weiss, Mohammad Norouzi, and William Chan. 2021. WaveGrad: Estimating gradients for waveform generation. In *Proceedings of the International Conference on Learning Representations (ICLR'21).*

Silvia Chiappa, Sébastien Racanière, Daan Wierstra, and Shakir Mohamed. 2017. Recurrent environment simulators. *arXiv* (2017).

Cassidy Curtis, Sigurdur Orn Adalgeirsson, Horia Stefan Ciurdar, Peter McDermott, J. D. Velásquez, W. Bradley Knox, Alonso Martinez, Dei Gaztelumendi, Norberto Adrian Goussies, Tianyu Liu, and Palash Nandy. 2022. Toward believable acting for autonomous animated characters. In *Proceedings of the 15th ACM SIGGRAPH Conference on Motion, Interaction and Games.*

Rishabh Dabral, Muhammad Hamza Mughal, Vladislav Golyanik, and Christian Theobalt. 2023. MoFusion: A framework for denoising-diffusion-based motion synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'23).*

Aram Davtyan and Paolo Favaro. 2022. Controllable video generation through global and local motion dynamics. In *Proceedings of the European Conference of Computer Vision (ECCV'22).*

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. 2022. Continuous diffusion for categorical data. https://doi.org/10.48550/arXiv.2211.15089

Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. 2020. GP-VAE: Deep probabilistic time series imputation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics.* PMLR.

Maurice Fréchet. 1957. Sur la distance de deux lois de probabilité. *Comptes Rendus Hebdom. Seances Acad. Sci.* 244, 6 (1957), 689–692.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance fields without neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'22).*

Tsu-Jui Fu, Licheng Yu, Ning Zhang, Cheng-Yang Fu, Jong-Chyi Su, William Yang Wang, and Sean Bell. 2023. Tell me what happened: Unifying text-guided video completion via multimodal masked video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'23).*

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2013. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13).*

Jason Gregory. 2018. *Game Engine Architecture.* CRC Press.

Ligong Han, Jian Ren, Hsin-Ying Lee, Francesco Barbieri, Kyle Olszewski, Shervin Minaee, Dimitris Metaxas, and Sergey Tulyakov. 2022. Show me what and tell me how: Video synthesis via multimodal conditioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'22).*

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS'17).*

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. 2022. *Imagen Video: High Definition Video Generation with Diffusion Models.* https://doi.org/10.48550/arXiv.2210.02303

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS'20).*

Jonathan Ho, Tim Salimans, Alexey A. Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. 2022b. Video diffusion models. In *Proceedings of the ICLR Workshop on Deep Generative Models for Highly Structured Data.*

Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. 2020. Learned motion matching. *ACM Trans. Graph.* 39, 40 (2020).

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Trans. Graph.* 36, 4 (2017).

Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. 2022. *CogVideo: Large-scale Pretraining for Text-to-Video Generation via Transformers.* https://doi.org/10.48550/arXiv.2205.15868

Jiahui Huang, Yuhe Jin, Kwang Moo Yi, and Leonid Sigal. 2022. Layered controllable video generation. In *Proceedings of the European Conference of Computer Vision (ECCV'22).*

Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. 2022. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22).*

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference of Computer Vision (ECCV'16).*

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of StyleGAN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'20).*

Seung Wook Kim, Jonah Philion, Antonio Torralba, and Sanja Fidler. 2021. DriveGAN: Towards a controllable high-quality neural simulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'21).*

Seung Wook Kim, Yuhao Zhou, Jonah Philion, Antonio Torralba, and Sanja Fidler. 2020. Learning to simulate dynamic environments with GameGAN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'20).*

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, (ICLR'15)*, San Diego, CA, Conference Track Proceedings. Retrieved from http://arxiv.org/abs/1412.6980

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2020. DiffWave: A versatile diffusion model for audio synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR'20).*

Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J. Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas A. Funkhouser. 2022. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'22).*

Yong-Hoon Kwon and Min-Gyu Park. 2019. Predicting future frames using retrospective cycle GAN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19).*

Max W. Y. Lam, Jun Wang, Dan Su, and Dong Yu. 2022. BDDM: Bilateral denoising diffusion models for fast and high-quality speech synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR'22).*

Kyungho Lee, Seyoung Lee, and Jehee Lee. 2018. Interactive character animation by learning multi-objective control. *ACM Trans. Graph.* 37, 6 (2018).

Yichong Leng, Zehua Chen, Junliang Guo, Haohe Liu, Jiawei Chen, Xu Tan, Danilo Mandic, Lei He, Xiangyang Li, Tao Qin, sheng zhao, and Tie-Yan Liu. 2022. BinauralGrad: A two-stage conditional diffusion probabilistic model for binaural audio synthesis. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS'22).*

J. P. Lewis, Matt Cordner, and Nickson Fong. 2000. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the SIGGRAPH Conference.*

Ruilong Li, Julian Tanke, Minh Vo, Michael Zollhofer, Jurgen Gall, Angjoo Kanazawa, and Christoph Lassner. 2022. TAVA: Template-free animatable volumetric actors. In *Proceedings of the European Conference of Computer Vision (ECCV'22).*

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. 2023. Magic3D: High-resolution text-to-3D content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'23).*

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. RoBERTa: A robustly optimized BERT pretraining approach. In *Proceedings of the International Conference on Learning Representations (ICLR'20).*

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A skinned multi-person linear model. *ACM Trans. Graph.* 34, 6 (2015).

Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. 2021. Acorn: Adaptive coordinate networks for neural scene representation. *ACM Trans. Graph.* 40, 4 (2021).

Willi Menapace, Stephane Lathuiliere, Sergey Tulyakov, Aliaksandr Siarohin, and Elisa Ricci. 2021. Playable video generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'21).*

Willi Menapace, Stéphane Lathuilière, Aliaksandr Siarohin, Christian Theobalt, Sergey Tulyakov, Vladislav Golyanik, and Elisa Ricci. 2022. Playable environments: Video manipulation in space and time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'22).*

Chenlin Meng, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. 2022. On distillation of guided diffusion models. In *Proceedings of the NeurIPS 2022 Workshop on Score-based Methods.*

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'19).*

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference of Computer Vision (ECCV'20).*

Norman Müller, Andrea Simonelli, Lorenzo Porzi, Samuel Rota Bulò, Matthias Nießner, and Peter Kontschieder. 2022. AutoRF: Learning 3D object radiance fields from single view observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'22).*

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* (2022). Retrieved from https://arxiv.org/abs/2201.05989

Thomas Müller, Fabrice Rousselle, Alexander Keller, and Jan Novák. 2020. Neural control variates. *ACM Trans. Graph.* 39, 6 (2020).

Michael Niemeyer and Andreas Geiger. 2021. GIRAFFE: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'21).*

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L. Lewis, and Satinder Singh. 2015. Action-conditional video prediction using deep networks in Atari games.

In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS'15)*.

Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. 2021. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'21)*.

Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. 2021a. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'21)*.

Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. 2021b. HyperNeRF: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.* 40, 6 (2021).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2022. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21, 1 (2022).

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. https://doi.org/10.48550/arXiv.2204.06125

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS'15)*.

ReplayMod. 2022. ReplayMod. Retrieved from https://github.com/ReplayMod/ReplayMod

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2021. High-resolution image synthesis with latent diffusion models. *arXiv* (2021).

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of the Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'15)*.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. 2022. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*. Retrieved from https://openreview.net/forum?id=08Yk-n5l2Al

Tim Salimans and Jonathan Ho. 2022. Progressive distillation for fast sampling of diffusion models. In *Proceedings of the International Conference on Learning Representations (ICLR'22)*.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W. Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R. Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. 2022. LAION-5B: An open large-scale dataset for training next generation image-text models. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS'22) Datasets and Benchmarks Track*.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. 2023. Make-A-Video: Text-to-video generation without text-video data. In *The Eleventh International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=nJfylDvgzlq

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising diffusion implicit models. In *Proceedings of the International Conference on Learning Representations (ICLR'21)*.

Matt Stanton, Sascha Geddert, Adrian Blumer, Paul Hormis, Andy Nealen, Seth Cooper, and Adrien Treuille. 2016. Large-scale finite state game engines. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation*.

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural state machine for character-scene interactions. *ACM Trans. Graph.* 38, 6 (2019).

Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. 2020. Local motion phases for learning multi-contact character movements. *ACM Trans. Graph.* 39, 4 (2020).

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS'21)*.

Guy Tevet, Brian Gordon, Amir Hertz, Amit H. Bermano, and Daniel Cohen-Or. 2022. MotionCLIP: Exposing human motion generation to CLIP space. In *Proceedings of the European Conference of Computer Vision (ECCV'22)*.

Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. 2021. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'21)*.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. 2018. Towards accurate generative models of video: A new metric & challenges. CoRR abs/1812.01717, (2018). Retrieved from http://arxiv.org/abs/1812.01717

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NeurIPS'17)*.

Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. 2022. HumanNeRF: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'22)*.

Yinghao Xu, Menglei Chai, Zifan Shi, Sida Peng, Ivan Skorokhodov, Aliaksandr Siarohin, Ceyuan Yang, Yujun Shen, Hsin-Ying Lee, Bolei Zhou, and Sergey Tulyakov. 2023. DisCoScene: Spatially disentangled generative radiance fields for controllable 3D-aware scene synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'23)*. 4402–4412.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. PlenOctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'21)*.

Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. 2022. MotionDiffuse: Text-driven human motion generation with diffusion model. https://doi.org/10.48550/arXiv.2208.15001

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*.

Received 2 May 2023; revised 30 September 2023; accepted 13 November 2023