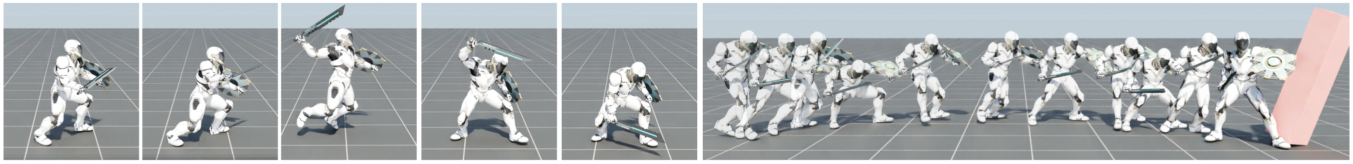
PADL: Language-Directed Physics-Based Character Control

Jordan Juravsky
NVIDIA
University of Waterloo
Canada
jjuravsky@nvidia.com

Yunrong Guo
NVIDIA
Canada
kellyg@nvidia.com

Sanja Fidler
NVIDIA
University of Toronto
Canada
sfidler@nvidia.com

Xue Bin Peng
NVIDIA
Simon Fraser University
Canada
japeng@nvidia.com



(a) Skill command: "jump and swing sword down".

(b) Skill command: "shield charge forward".

Figure 1: Our framework allows users to direct the behaviors of physically simulated characters using natural language commands. Left: Humanoid character performing a jump attack. Right: Character knocking over a target object by performing a shield charge.

ABSTRACT

Developing systems that can synthesize natural and life-like motions for simulated characters has long been a focus for computer animation. But in order for these systems to be useful for downstream applications, they need not only produce high-quality motions, but must also provide an accessible and versatile interface through which users can direct a character's behaviors. Natural language provides a simple-to-use and expressive medium for specifying a user's intent. Recent breakthroughs in natural language processing (NLP) have demonstrated effective use of language-based interfaces for applications such as image generation and program synthesis. In this work, we present PADL, which leverages recent innovations in NLP in order to take steps towards developing language-directed controllers for physics-based character animation. PADL allows users to issue natural language commands for specifying both high-level tasks and low-level skills that a character should perform. We present an adversarial imitation learning approach for training policies to map high-level language commands to low-level controls that enable a character to perform the desired task and skill specified by a user's commands. Furthermore, we propose a multi-task aggregation method that leverages a language-based multiple-choice question-answering approach to determine high-level task objectives from language commands. We show that our framework can be applied to effectively direct a simulated humanoid character to perform a diverse array of complex motor skills.

CCS CONCEPTS

• **Computing methodologies → Procedural animation**; *Adversarial learning*.

KEYWORDS

character animation, language commands, reinforcement learning, adversarial imitation learning

ACM Reference Format:
Jordan Juravsky, Yunrong Guo, Sanja Fidler, and Xue Bin Peng. 2022. PADL: Language-Directed Physics-Based Character Control. In *SIGGRAPH Asia 2022 Conference Papers (SA '22 Conference Papers), December 6–9, 2022, Daegu, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3550469.3555391

1 INTRODUCTION

Developing physically simulated characters that are capable of producing complex and life-like behaviors has been one of the central challenges in computer animation. Efforts in this domain has led to systems that can produce high-quality motions for a wide range of skills [Clegg et al. 2018; de Lasa et al. 2010; Hodgins et al. 1995; Lee et al. 2010a; Liu and Hodgins 2018; Liu et al. 2016; Mordatch et al. 2012; Peng et al. 2018a; Tan et al. 2014; Wang et al. 2009]. However, in order for these systems to be useful for downstream applications, the control models need not only produce high quality motions, but also provide users with an accessible and versatile interface through which to direct a character's behaviors. This interface is commonly instantiated through compact control abstractions, such as joystick controls or target way points. These control abstractions allow users to easily direct a character's behavior via high-level commands, but they can greatly restrict the variety and granularity of the behaviors that a user can actively control. Alternatively, motion tracking models can provide a versatile interface that enables fine-grain control over a character's movements by directly specifying target motion trajectories. However, authoring motion trajectories can be a labour-intensive process, requiring significant domain expertise or specialized equipment (e.g. motion capture).

SA '22 Conference Papers, December 6–9, 2022, Daegu, Republic of Korea

ACM ISBN 978-1-4503-9470-3/22/12…$15.00
https://doi.org/10.1145/3550469.3555391

An ideal animation system should provide an accessible interface that allows users to easily specify desired behaviors for a character, while also being sufficiently versatile to enable control over a rich corpus of skills. Natural language offers a promising medium that is both accessible and versatile. The recent development of large and expressive language models has provided powerful tools for integrating natural language interfaces for a wide range of downstream applications [Brown et al. 2020; Devlin et al. 2018; Radford et al. 2021], such as generating functional code and realistic images from natural language descriptions [Chen et al. 2021; Ramesh et al. 2022; Tan et al. 2018]. In this work, we aim to leverage these techniques from NLP to take steps towards developing a language-directed system for physics-based character animation.

The central contribution of this work is a system for language-directed physics-based character animation, which enables users to direct the behaviors of a physically simulated character using natural language commands. Given a dataset of motion clips and captions, which describe the behaviors depicted in each clip, our system trains control policies to map from high-level language commands to low-level motor commands that enable a character to reproduce the corresponding skills. We present an adversarial imitation learning approach that allows a policy to reproduce a diverse array of skills, while also learning to ground each skill in language commands. Our policies can also be trained to perform additional auxiliary tasks. We present a language-based multi-task aggregation model, which selects between a collection of task-specific policies according to a given command, thereby allowing users to easily direct a character to perform various high-level tasks via natural language. We present one of the first systems that can effectively leverage language commands to direct full-body physically simulated character to perform a diverse array of complex motor skills. The code for this work is available at https://github.com/nv-tlabs/PADL.

2 RELATED WORK

Synthesizing natural and intelligent behaviors for simulated characters has been a core subject of interest in computer animation, with a large body of work focused on building kinematic and physics-based control models that can generate life-like motions [Clegg et al. 2018; da Silva et al. 2008; Hodgins et al. 1995; Holden et al. 2016; Lee et al. 2010a; Liu and Hodgins 2018; Tan et al. 2014; Wang et al. 2009, 2012]. While a great deal of emphasis has been placed on motion quality, considerably less attention has been devoted on the *directability* of the resulting models at run-time. Directability is often incorporated into these models via control abstractions that allow users to direct a character's behaviors through high-level commands. These abstractions tend to introduce a trade-off between accessibility and versatility. Simple control abstractions, such as joystick commands or target waypoints, [Agrawal and van de Panne 2016; Coros et al. 2009; Holden et al. 2017; Lee et al. 2021b,a, 2010b; Ling et al. 2020; Peng et al. 2018a, 2022, 2021; Starke et al. 2019; Treuille et al. 2007; Zhang et al. 2020], provide an accessible interface that can be easily adopted by users. But these abstractions can also limit the versatility of the behaviors that can be actively controlled by a user. Alternatively, general motion tracking models can provide a versatile interface, which allows for fine-grain control over a character's movements through target motion trajectories [Bergamin et al. 2019; Park et al. 2019; Pollard et al. 2002; Wang et al. 2020; Won et al. 2020; Yamane et al. 2010]. These target trajectories specify desired poses for the character to reach at every timesteps, which in principle can direct the character to perform any feasible motion. However, this versatility often comes at the cost of accessibility, since authoring target motion trajectories can be as tedious and labour intensive as manual keyframe animation. Motion capture can be a more expeditious approach for generating target trajectories for motion-tracking models [Peng et al. 2018b; Wang et al. 2020; Yu et al. 2021; Yuan et al. 2021], but tends to require specialized equipment and may limit the reproducible behaviors to those that can be physically performed by the user. In this work, we aim to leverage natural language to develop an accessible and versatile control interface for physics-based character animation.

Natural Language Processing: Language models trained on increasingly large datasets have been shown to develop powerful representations for text data [Devlin et al. 2018; Liu et al. 2019; Raffel et al. 2019], which can be used for a wide range of downstream applications. One such example is text-guided synthesis, where a user's prompt, expressed in natural language, can be used to direct models to produce different types of content. Large autoregressive models are able to generate coherent text completions given a user's starter prompt [Brown et al. 2020]. These models lead to the popularization of "prompt engineering", where the aim is to construct optimal prompt templates that elicit the desired behaviors from a language model. Such prompt-based systems, often combined with filtering or other post-processing techniques, have been successfully used to solve grade-school math problems and competitive programming challenges [Cobbe et al. 2021; Li et al. 2022]. Text-guided synthesis can also be applied across different modalities. Here, the language model does not directly generate the desired content, instead it provides a semantically meaningful encoding for a user's language prompt, which can then be used by a separately trained decoder to generate content in a different modality. Nichol et al. [2021] and Ramesh et al. [2022] successfully used this approach to generate photo-realistic images from natural language, leveraging the text encoder from CLIP [Radford et al. 2021]. In this work, we aim to leverage powerful language models to develop language-directed controllers for physics-based character animation.

Language-Directed Animation: Synthesizing motion from language is one of the core challenges of audio-driven facial animation, where the goal is to generate facial motions for a given utterance. These models typically take advantage of the temporal correspondence between units of speech (phonemes) and facial articulations (visemes) in order to synthesize plausible facial animations for a particular utterance [Brand 1999; Deena and Galata 2009; Hong et al. 2002; Karras et al. 2017; Pelachaud et al. 1996]. A similar temporal correspondence can also be leveraged to generate full-body gestures from speech [Ahuja and Morency 2019; Alexanderson et al. 2020; Levine et al. 2009]. While these techniques can be highly effective for generating realistic motions from speech, they are not directly applicable in more general settings where there is no clear temporal correspondence between language and motion. For example, a high-level command such as "knock over the red block" implicitly encodes a sequence of skills that a character should perform. Sequence-to-sequence models have been proposed to map high-level language descriptions to motion trajectories [Lin et al. 2018; Plappert et al. 2017]. Ahuja and Morency [2019] and Tevet et al. [2022] proposed autoencoder frameworks that learns a joint embedding of language and motion, which can be used to generate full-body motions from language descriptions. While these techniques have demonstrated promising results, they have been primarily focused on developing kinematic motion models. In this work, we aim to develop a language-directed model for physics-based character animation, which maps high-level language commands to low-level controls that enable a character to perform the desired behaviors.

3 BACKGROUND

Our characters are trained using a goal-conditioned reinforcement learning framework, where an agent interacts with an environment according to a control policy π in order to fulfill a given goal $\mathbf{g} \in \mathcal{G}$, drawn from a goal distribution $\mathbf{g} \sim p(\mathbf{g})$. At each time step t, the agent observes the state of the environment $\mathbf{s}_t \in \mathcal{S}$, and responds by applying an action $\mathbf{a}_t \in \mathcal{A}$, sampled from the policy $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{g})$. After applying the action $\mathbf{a}_t$, the environment transitions to a new state $\mathbf{s}_{t+1}$, and the agent receives a scalar reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{g})$ that reflects the desirability of the state transition for the given goal $\mathbf{g}$. The agent's objective is to learn policy π that maximizes its expected discounted return $J(\pi)$,

$$J(\pi) = \mathbb{E}_{p(\mathbf{g})}\mathbb{E}_{p(\tau|\pi,\mathbf{g})}\left[\sum_{t=0}^{T-1}\gamma^t r_t\right], \tag{1}$$

where $p(\tau|\pi, \mathbf{g}) = p(\mathbf{s}_0)\prod_{t=0}^{T-1} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\pi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{g})$ denotes the likelihood of a trajectory $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, ..., \mathbf{s}_T)$ under a policy π given a goal $\mathbf{g}$, $p(\mathbf{s}_0)$ is the initial state distribution, and $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{s}_a)$ represents the transition dynamics of the environment. T is the time horizon of a trajectory, and $\gamma \in [0, 1]$ is a discount factor.

4 OVERVIEW

In this paper we introduce Physics-based Animation Directed with Language (PADL; pronounced "paddle"), a system for developing language-directed control models for physics-based character animation. Our framework allows users to control the motion of a character by specifying a *task* to complete, as well as a specific *skill* to use while completing that task. Tasks represent high-level objectives that the agent must accomplish, such as navigating to a target location or interacting with a specific object. In addition to specifying *what* task an agent must accomplish, it is important for users to be able to control *how* the task is accomplished. For example, given the task of navigating to a target location, an agent can walk, run, or jump to the target. In our system, the desired task and skill for the character are specified separately via natural language in the form of a task command and a skill command.

Our framework consists of three stages, and a schematic overview of the system is available in Figure 2. First, in the *Skill Embedding* stage, a reference motion dataset $\mathcal{M} = \{(\mathbf{m}^i, c^i)\}$, containing motion clips $\mathbf{m}^i$ annotated with natural language captions c^i, is used to learn a shared embedding space $\mathcal{Z}$ of motions and text. Each motion clip $m^i = \{\hat{\mathbf{q}}^i_t\}$ is represented by a sequence of poses $\hat{\mathbf{q}}^i_t$. A motion encoder $z^i_m = \text{Enc}_m(\mathbf{m}^i)$ and language encoder $z^i_l = \text{Enc}_l(c^i)$ are trained to map each motion and caption pair to similar embeddings $z^i_m \approx z^i_l$. Next, in the *Policy Training* stage, this embedding is used to train a collection of reinforcement learning policies, where each policy $\pi^i(\mathbf{a}_t|\mathbf{s}_t, \mathbf{g}, \mathbf{z})$ is trained to perform a particular task using various skills $\mathbf{z} \in \mathcal{Z}$ from the embedding. Once trained, the policy can then be directed to execute a particular skill by conditioning π on the embedding of a given language command $z_l = \text{Enc}_l(c)$. Finally, in the *Multi-Task Aggregation* stage, the different policies are integrated into a multi-task controller that can be directed using language commands to perform a specific task using a desired skill.

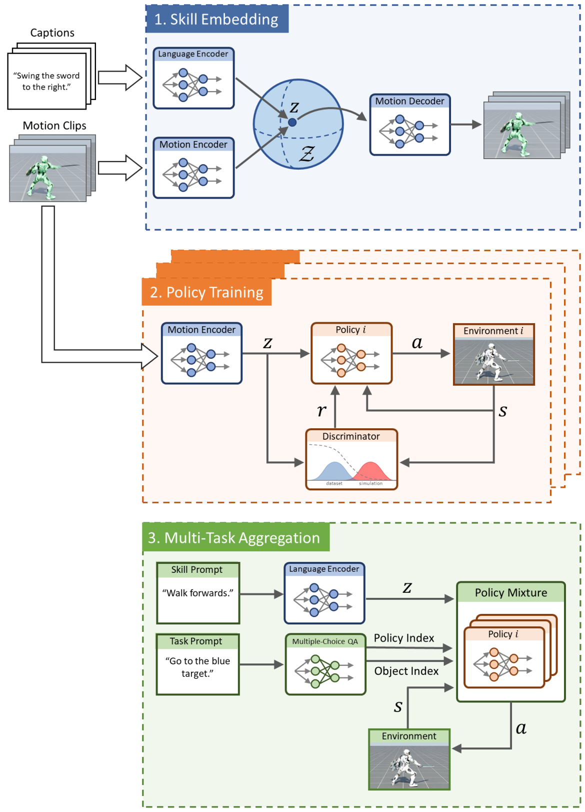


Figure 2: The PADL framework consists of three stages. 1) In the Skill Embedding stage, a dataset of motion clips and corresponding text captions are used to learn a joint embedding of motions and captions. 2) In the Policy Training stage, the learned skill embedding is used to train a collection of policies to perform various tasks, while imitating behaviors in the dataset. 3) Finally, in the Multi-Task Aggregation stage, policies trained for different tasks are combined into a multi-task controller that can be directed to perform different tasks and skills via language commands.

5 SKILL EMBEDDING

In the Skill Embedding stage, our objective is to construct an embedding space that aligns motions with their corresponding natural language descriptions. To do this, we follow a similar procedure as MotionCLIP [Tevet et al. 2022], where a transformer autoencoder is trained to encode motion sequences into a latent representation that "aligns" with the language embedding from a pre-trained CLIP text encoder [Radford et al. 2021]. Given a motion clip $\hat{\mathbf{m}} = (\hat{\mathbf{q}}_1, ..., \hat{\mathbf{q}}_n)$ and its caption c, a motion encoder $\mathbf{z} = \mathrm{Enc}_m(\hat{\mathbf{m}})$ maps the motion to an embedding $\mathbf{z}$. The embedding is normalized to lie on a unit sphere $||\mathbf{z}|| = 1$. Following Tevet et al. [2022], $\mathrm{Enc}_m(\mathbf{m})$ is modeled by a bidirectional transformer [Devlin et al. 2018]. A motion decoder is jointly trained with the encoder to produce a reconstruction sequence $\mathbf{m} = (\mathbf{q}_1, ..., \mathbf{q}_n)$ to recover $\hat{\mathbf{m}}$ from $\mathbf{z}$. The decoder is also modelled as a birectional transformer $\mathbf{m} = \mathrm{Dec}(\mathbf{z}, \mathbf{U})$, which decodes all frames of in parallel using a learned constant query sequence $\mathbf{U} = (\mathbf{u}_1, ..., \mathbf{u}_n)$, similar to the final layer of Carion et al. [2020]. The autoencoder is trained with the loss:

$$\mathcal{L}_{\mathrm{auto}} = \mathcal{L}_{\mathrm{recon}} + 0.1\mathcal{L}_{\mathrm{align}}. \tag{2}$$

The reconstruction loss $\mathcal{L}_{\mathrm{recon}}$ measures the error between the reconstructed sequence and original motion:

$$\mathcal{L}_{\mathrm{recon}} = \frac{1}{n}\sum_{t=1}^{n} ||\hat{\mathbf{q}}_t - \mathrm{Dec}\left(\mathrm{Enc}_m\left(\hat{\mathbf{m}}\right), \mathbf{U}\right)||_2^2. \tag{3}$$

The alignment loss $\mathcal{L}_{\mathrm{align}}$ measures the cosine distance between a motion embedding and the language embedding:

$$\mathcal{L}_{\mathrm{align}} = 1 - d_{\cos}\left(\mathrm{Enc}_m\left(\hat{\mathbf{m}}\right), \mathrm{Enc}_l(c)\right). \tag{4}$$

The language encoder $\mathrm{Enc}_l\left(\mathbf{m}\right)$ is modeled using a pre-trained CLIP text encoder with an added head of two fully-connected layers, where only this output head is fine-tuned according to Eq. 4. To help avoid overfitting, for every minibatch of motion sequences sampled from the dataset we also extract a random subsequence from each motion and add these slices to the batch that the model is trained on. These subsequences only contribute to the reconstruction loss.

6 POLICY TRAINING

Once we have a joint embedding of motions and captions, we will next use the embedding to train control policies that enable a physically simulated character to perform various high-level tasks while using skills specified by language commands. At each timestep t, the policy $\pi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{g}, \mathbf{z})$ receives as input the state of the character $\mathbf{s}_t$, a task-specific goal $\mathbf{g}$, and a skill latent $\mathbf{z}$. The goal $\mathbf{g}$ specifies high-level task objectives that the character should achieve, such as moving to a target location or facing a desired direction. The skill latent $\mathbf{z}$ specifies the skill that the character should use to achieve the desired goal, such as walking vs running to a target location. The latents are generated by encoding motion clips $\mathbf{z} = \mathrm{Enc}_m(\mathbf{m})$ sampled from the dataset $\mathcal{M}$. In order to train a policy to perform a given task using a desired skill, we utilize a reward function consisting of two components:

$$r_t = r_t^{\mathrm{skill}} + \lambda^{\mathrm{task}} r_t^{\mathrm{task}}, \tag{5}$$

where r_t^{skill} is a skill-reward, and r_t^{task} is a task-reward with coefficient λ^{task}.

6.1 Skill Objective

To train the policy to perform the skill specified by a particular $\mathbf{z}_i$, we enforce that the policy's distribution of state transitions $(\mathbf{s}, \mathbf{s}')$ matches that of the corresponding motion clip $\mathbf{m}^i$. To accomplish this, we train an adversarial discriminator $D(\mathbf{s}, \mathbf{s}', \mathbf{z})$ on the joint distribution of state transitions and skill encodings [Ho and Ermon 2016; Merel et al. 2017; Peng et al. 2021]. The discriminator is trained to predict if a given state transition $(\mathbf{s}, \mathbf{s}')$ is from the motion clip corresponding to $\mathbf{z}$, or if the transition is from the simulated character or from a different motion clip in the dataset. The discriminator is trained by minimizing the following loss:

$$\mathcal{L}_D = \mathbb{E}_{p_{\mathcal{M}}(\mathbf{m})}\Bigg[-\mathbb{E}_{p_{\mathbf{m}}(\mathbf{s},\mathbf{s}')}\left[\log(D(\mathbf{s}, \mathbf{s}', \mathbf{z}))\right] \tag{6}$$

$$- w_D\, \mathbb{E}_{p_{\pi}(\mathbf{s},\mathbf{s}'|\mathbf{z})}\left[\log(1 - D(\mathbf{s}, \mathbf{s}', \mathbf{z}))\right] \tag{7}$$

$$- (1 - w_D)\, \mathbb{E}_{p_{\mathcal{M}\backslash\mathbf{m}}(\mathbf{s},\mathbf{s}')}\left[\log(1 - D(\mathbf{s}, \mathbf{s}', \mathbf{z}))\right] \tag{8}$$

$$+ w_{\mathrm{gp}}\, \mathbb{E}_{p_{\mathbf{m}}(\mathbf{s},\mathbf{s}')}\left[\left\|\nabla_\phi D(\phi, \mathbf{z})\Big|_{\phi=(\mathbf{s},\mathbf{s}')}\right\|^2\right]\Bigg]. \tag{9}$$

$p_{\mathcal{M}}(\mathbf{m})$ represents the likelihood of sampling a motion clip $\mathbf{m}$ from a dataset $\mathcal{M}$, and $\mathbf{z} = \mathrm{Enc}_m(\mathbf{m})$ is the encoding of the motion clip. $p_{\mathbf{m}}(\mathbf{s}, \mathbf{s}')$ denotes the likelihood of observing a state transition from a given motion clip, and $p_\pi(\mathbf{s}, \mathbf{s}'|\mathbf{z})$ is the likelihood of observing a state transition from the policy π when conditioned on $\mathbf{z}$. $p_{\mathcal{M}\backslash\mathbf{m}}(\mathbf{s}, \mathbf{s}')$ represents the likelihood of observing a state transition by sampling random transitions from other motion clips in the dataset, excluding $\mathbf{m}$, and w_D is a manually specified coefficient. The final term in the loss is a gradient penalty with coefficient w_{gp} [Peng et al. 2021], which improves stability of the adversarial training process. The skill-reward is then given by:

$$r_t^{\mathrm{skill}} = -\log\left(1 - D(\mathbf{s}_t, \mathbf{s}_{t+1}, \mathbf{z})\right). \tag{10}$$

To direct the policy with a skill command c_{skill} after it has been trained, the model is provided with the encoding $\mathbf{z} = \mathrm{Enc}_l(c_{\mathrm{skill}})$. By conditioning the discriminator on both state transitions and latents, our method explicitly encourages the policy to imitate every motion clip in the dataset, which can greatly reduce mode collapse. We elaborate on this benefit and compare our approach to related adversarial RL frameworks in Appendix D.

7 MULTI-TASK AGGREGATION

Each policy from the Policy Training stage is capable of performing a variety of skills, but each is only able to perform a single high-level task involving a single target object. We show that these individual policies can be aggregated into a more flexible composite policy, which allows users to direct the character to perform a variety of different tasks in an environment containing multiple objects. However, in our experiments, we found that attempting to use the procedure in Section 6 to train a single multi-task policy to perform all tasks leads to poor performance. Effectively training multi-task policies remains a challenging and open problem in RL, and prior systems have often taken a divide-and-conquer approach for multi-task RL [Ghosh et al. 2018; Ruder 2017; Rusu et al. 2015].

To create a more flexible multi-task, multi-object controller, we aggregate a collection of single-task policies together. At each timestep, the user's current task command is used to generate prompts that are fed to a multiple-choice question-answering (QA) model. The QA model identifies which task and environment object are being referenced by the user. The single-task controller for the identified task is then set as the active policy controlling the character, and the state of the identified object is passed to the selected policy. An overview of this procedure is provided with pseudocode in Algorithm 1 in the Appendix. Note that since the character is being driven by a single policy from Section 6 at every timestep, the aggregated controller can only follow one high-level task involving a single object at a time. However, with this controller the user can dynamically control which task and object are focused on using natural language.

7.1 Multiple Choice Question Answering

An overview of the language-based selection model is shown in Figure 3. The multiple-choice QA model is constructed using a pre-trained BERT model fine-tuned on the SWAG dataset [Zellers et al. 2018]. Each multiple-choice question is formulated as an initial prompt sentence (Sentence A) alongside n candidate follow-up sentences (Sentence B) [Devlin et al. 2018]. The model then outputs scores for n distinct sequences, where sequence i is the concatenation of the prompt sentence with the i-th candidate sentence. The object corresponding to the candidate sentence with the highest score is selected as the target object for the policy. A similar process is used to identify the task from the user’s command.

For each task command provided by the user, the model is provided with two separate multiple-choice questions to identify the relevant task and object, respectively. The first question identifies the task, where each multiple choice option corresponds to a trained policy. The inputs to the QA model follow a story-like format in order to mimic the elements of the SWAG dataset that the model was fine-tuned on. For example, if the task command is *“knock over the blue tower”*, the candidate sequence for the strike policy is:

- "Bob wants to *knock over the blue tower*. This should be easy for him since he possesses the ability to *knock over a specified object*."

Similarly, the candidate sequence for the location policy is given by:

- "Bob wants to *knock over the blue tower*. This should be easy for him since he possesses the ability to *navigate to a specified destination*."

The multiple-choice QA model will then predict which sequence of sentences are most likely. Similarly, in the multiple-choice question to extract the target object, each object is given a multiple choice option describing the object’s appearance. The candidate sequence for the green block is given by:

- "Bob wants to *knock over the blue tower*. He starts by turning his attention to *the green object* nearby."

8 EXPERIMENTAL SETUP

We evaluate the effectiveness of our framework by training language-directed control policies for a 3D simulated humanoid character. The character is equipped with a sword and shield, similar to the one used by Peng et al. [2022], with 37 degrees-of-freedom, and similar state and action representations. The dataset contains a total of 131 individual clips, for a total of approximately 9 minutes of motion data. Each clip is manually labeled with 1-4 captions that describe the behavior of the character within a particular clip, for a total of 265 captions in the entire dataset. Fig. 4 illustrates examples of motion clips in the dataset along with their respective captions.

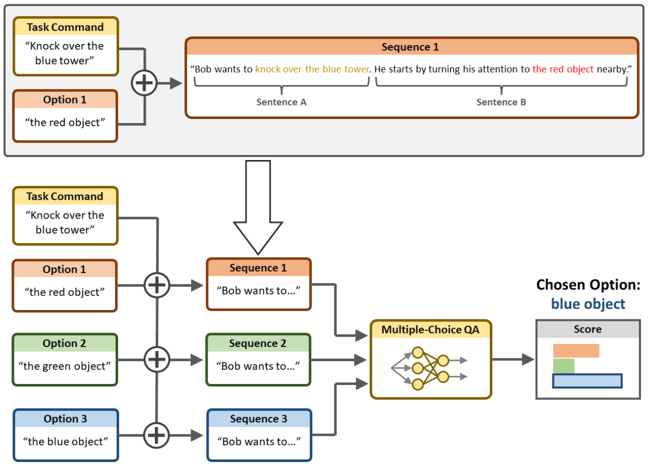


Figure 3: Overview of the language-based selection model used to select a target object based on the user’s task command. The task command is used to generate a collection of candidate sentences, each corresponding to a particular object in the environment. A multiple-choice QA model is then used to predict the most likely candidate sentence, based on the task command. The model’s prediction is used to identify the target object the user referenced.

8.1 Tasks

In addition to training policies to imitate skills from the dataset, each policy is also trained to perform an additional high-level task. Here, we provide an overview of the various tasks, and more detailed descriptions are available in Appendix B.

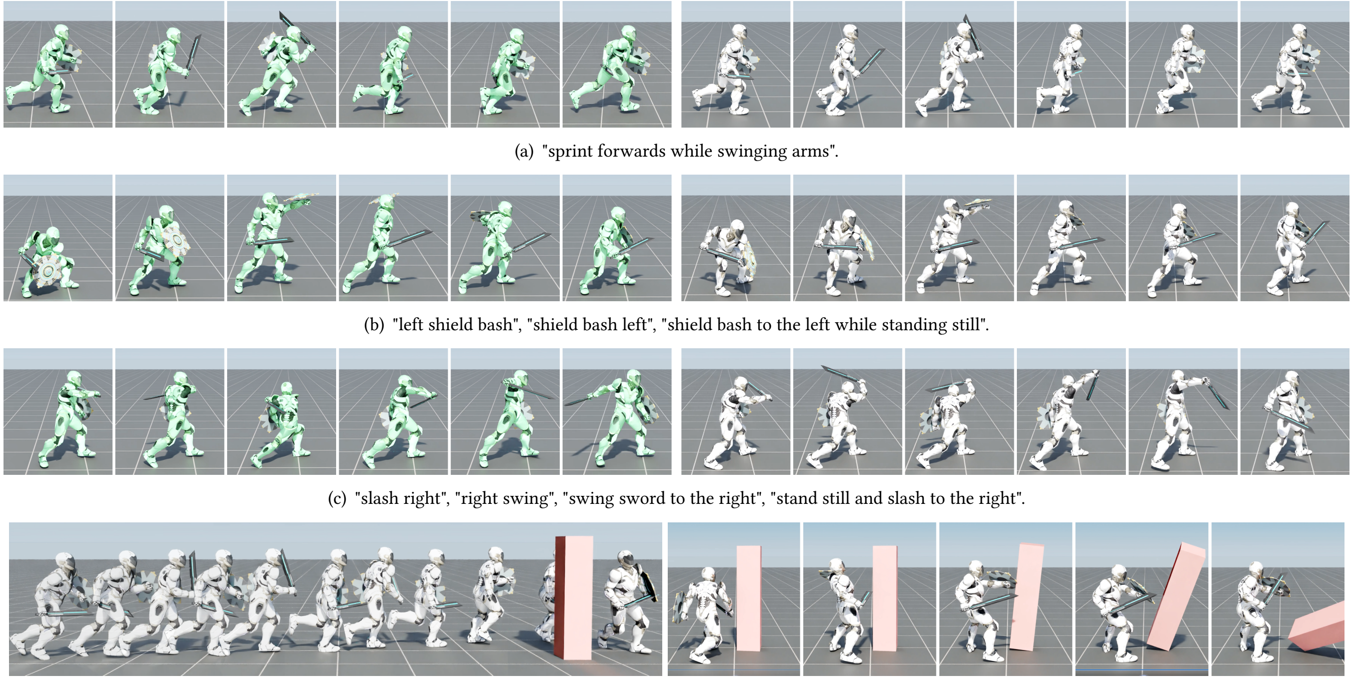
(1) **Facing:** First, we have a simple facing task, where the objective is for the character to turn and face a target direction $\mathbf{d}^*$, encoded as a 2D vector on the horizontal plane. The goal input $\mathbf{g}_t = \tilde{\mathbf{d}}^*_t$ for the policy records the goal direction in the character’s local coordinate frame.

(2) **Location:** Next, we have a target location task, where the objective is for the character to navigate to a target location $\mathbf{x}^*$. The goal $\mathbf{g}_t = \tilde{\mathbf{x}}^*_t$ records the target location in the character’s local coordinate frame $\tilde{\mathbf{x}}^*_t$.

(3) **Strike:** Finally, we have a strike task, where the objective is for the character to knock over a target object. The goal $\mathbf{g}_t = (\tilde{\mathbf{x}}^*_t, \tilde{\dot{\mathbf{x}}}^*_t, \tilde{q}^*_t, \tilde{\dot{q}}^*_t)$ records the target object’s position $\tilde{\mathbf{x}}^*_t$, rotation $\tilde{q}^*_t$, linear velocity $\tilde{\dot{\mathbf{x}}}^*_t$, and angular velocity $\tilde{\dot{q}}^*_t$. All features are expressed in the character’s local frame.

8.2 Training

All physics simulations are performed using Isaac Gym, a massively parallel GPU-based physics simulator [Makoviychuk et al. 2021]. The simulation is performed at a frequency of 120Hz, while the policies operate at a frequency of 30Hz. 4096 environments are simulated in parallel on a single A100 GPU. A 128D latent space is used for the skill embedding. The policy, value function, and discriminator are modeled using separate multi-layer perceptions



(a) "sprint forwards while swinging arms".

(b) "left shield bash", "shield bash left", "shield bash to the left while standing still".

(c) "slash right", "right swing", "swing sword to the right", "stand still and slash to the right".

(d) task: Location. skill: "sprint forward while swinging arms".

(e) task: Strike. skill: "shield bash to the right".

Figure 4: (a) – (c): Reference motion clips (left side) and their corresponding captions, along with motions produced by a simulated character when directed to perform the reference skills through language commands (right side). More reference motions and policy trajectories are shown in Fig. 7 in the Appendix. (d) – (e): Trained policies completing tasks with different skills.

with ReLU units and hidden layers containing $[1024, 1024, 512]$ units. Each policy is trained using proximal policy optimization with about 7 billion samples [Schulman et al. 2017], corresponding to approximately 7 years of simulated time, which requires about 2.5 days of real-world time. Selecting a weight λ^{task} for the task reward that effectively balances the task and skill reward can be challenging, and may require task-specific tuning. We therefore apply an adaptive method to dynamically adjust λ^{task} based on a target task-reward value [Mentzer et al. 2021]. More details are available in Appendix B.4.

9 RESULTS

We first train policies without auxiliary tasks to evaluate the model's ability to reproduce skills from a motion dataset. Examples of the policy's behaviors when given various skill commands are available in Fig. 4. The policy is able to follow a variety of language commands, ranging from locomotion skills, such as walking and running, to more athletic behaviors, such as sword swings and shield bashes. Since the language encoder is built on a large CLIP model [Radford et al. 2021], it exhibits some robustness to new commands, which were not in the dataset. For example, the model correctly performs a casual walking motion when prompted with: *"take a leisurely stroll"*, even though no captions in the dataset contained *"leisurely"* or phrased walking as *"taking a walk"*. However, due to the relatively small amount of captions used to train the encoder, the model can still produce incorrect behaviors for some new commands. The character successfully performs a right slash when given the prompt: *"right slash"*. However, *"right slash with sword"* leads the character to perform a left slash.

In addition to learning skills from a motion dataset, our policies can also be trained to perform additional high-level tasks, as outlined in Section 8.1. Examples of the tasks are available in Figure 4. Separate policies are trained for each task, which can then be integrated into a single multi-task controller that activates the appropriate policy given a task command. We demonstrate the effectiveness of the multi-task controller in an environment containing multiple objects that the character can interact with. The user can issue a task command for specifying the target object and the desired task that the character should perform. Our multiple-choice question-answering framework is able to consistently identify the correct task and target object from a user's commands. For example, given the command: *"knock over the blue block"*. the selection model correctly identifies the policy for the Strike task, and selects the blue block as the target. The selection model can also parse more unusual commands, such as "mosey on down to the maroon saloon", which correctly identifies the Location task and selects the red block. Despite the generalization capabilities of large language models, some commands can still lead to incorrect behaviors. More examples of task commands and the resulting behaviors from the model are available in Appendix C.

9.1 Dataset Coverage

To determine the impact of learning a skill embedding that aligns motions and text, we evaluate our model's ability to reproduce

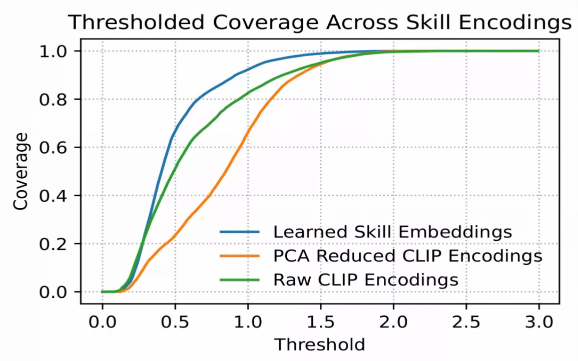


Figure 5: Comparing dataset coverage when different skill encodings are used during the Policy Training stage. “Learned Skill Embeddings” use the 128D embedding from the learned motion encoder detailed in Section 5. We compare against baselines where policies are trained directly using the 512D CLIP text encodings of the dataset captions and where these encodings are reduced to 128D using PCA.

various motions in the dataset when given the respective commands. We conduct this evaluation using a thresholded coverage metric. Given a sequence of states specified by a motion clip $\hat{\mathbf{m}} = (\hat{\mathbf{s}}_0, \hat{\mathbf{s}}_2, ..., \hat{\mathbf{s}}_n)$, a policy trajectory $\tau = (\mathbf{s}_0, \mathbf{s}_2, ..., \mathbf{s}_k)$ for a skill encoding $\mathbf{z} = \mathrm{Enc}_l(c)$ (where c is a caption for $\hat{\mathbf{m}}$), and a threshold parameter $\epsilon > 0$, we define the coverage to be:

$$\mathrm{coverage}(\tau, \hat{\mathbf{m}}, c, \epsilon) = \frac{1}{n}\sum_{i=0}^{n} \mathcal{I}\left(\left(\min_{j\in\{0,...,k\}} ||\hat{\mathbf{s}}_i - \mathbf{s}_j||_2\right) \leq \epsilon\right) \quad (11)$$

This metric determines the fraction of the states in a motion clip that are sufficiently close to a state in the policy’s trajectory. In our experiments we collect 300 timesteps (10 seconds) per trajectory. Instead of selecting a fixed threshold ϵ, we apply Equation 11 with different values of ϵ between $[0, 3]$ to produce a coverage curve.

Figure 5 compares the performance of the PADL model with baseline models that directly use the CLIP encoding of a caption as input to the policy. Coverage statistics are averaged across all the captions for each motion clip in the dataset, and then averaged across all motion clips. The raw CLIP encoding is 512D, while our learned skill embedding is 128D. We include an additional baseline model, which uses PCA to reduce the dimensionality of the CLIP encoding to 128D. Our learned embedding is able to better reproduce behaviors in the dataset. Directly using the CLIP encoding as input to the policy tends to result in lower quality motions, and has a higher tendency of performing incorrect behaviors when directed with language commands.

9.2 Skill Interpolation

In addition to enabling language control, the learned skill embedding also leads to semantically meaningful interpolations between different skills. Given two skill commands c_1 and c_2, we encode each caption into the corresponding latents $\mathbf{z}_1$ and $\mathbf{z}_2$ using the language encoder. We then interpolate between the two latents using spherical interpolation, and condition the policy on the interpolated latent to produce a trajectory. For example, given two commands: *“walk forward”* and *“sprint forward while swinging arms”*, interpolating between the two latents leads to locomotion behaviors that travel at different speeds. Figure 6 records the average velocity of the character when the policy is conditioned on different interpolated latents. Similarly, interpolating between *“walk forward”* and *“crouching walk forward”* leads to gaits with different walking heights. However, not all pairs of commands lead to intuitive intermediate behaviors.

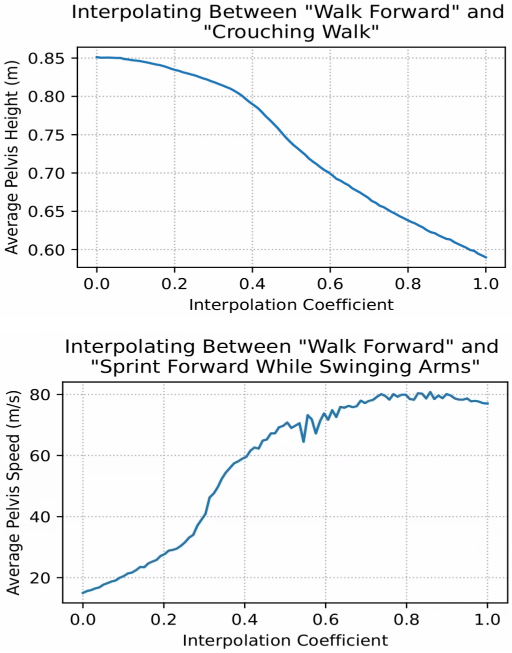


Figure 6: Interpolating skills in the latent space leads to semantically meaningful intermediate behaviors, such as traveling with different walking heights and speeds.

10 CONCLUSIONS

In this work we presented PADL, a framework for learning language-directed controllers for physics-based character animation. Language is used to specify both high-level tasks that a character should perform and low-level skills that the character should use to accomplish the tasks. While our models are able to imitate a diverse array of skills from motion data, the models remain limited in the variety of high-level tasks that they can perform. We are interested in exploring more scalable approaches to modelling character interactions with the environment, replacing the finite *a priori* collection of tasks with a more general strategy that allows the user to specify arbitrary environment interactions with natural language. We are additionally interested in scaling PADL to much larger labelled motion capture datasets [Punnakkal et al. 2021], which may lead to agents and language encoders that can model a greater diversity of skills while being more robust to paraphrasing and capable of generalizing to new commands. In particular, we expect the language encoder from the Skill Embedding stage to improve significantly with more text data. We are excited for further advances in language-guided physics-based character animation and hope that our work contributes towards the development of powerful, high-quality animation tools with broadly accessible, versatile, and easy-to-use interfaces.

ACKNOWLEDGMENTS

We would like to thank Reallusion[1] for providing motion capture reference data for this project. Additionally, we would like to thank the anonymous reviews for their feedback, and Steve Masseroni and Margaret Albrecht for their help in producing the supplementary video.

REFERENCES

Shailen Agrawal and Michiel van de Panne. 2016. Task-based Locomotion. *ACM Transactions on Graphics (Proc. SIGGRAPH 2016)* 35, 4 (2016).

C. Ahuja and L. Morency. 2019. Language2Pose: Natural Language Grounded Pose Forecasting. In *2019 International Conference on 3D Vision (3DV)*. IEEE Computer Society, Los Alamitos, CA, USA, 719–728. https://doi.org/10.1109/3DV.2019.00084

Simon Alexanderson, Gustav Eje Henter, Taras Kucherenko, and Jonas Beskow. 2020. Style-Controllable Speech-Driven Gesture Synthesis Using Normalising Flows. *Computer Graphics Forum* (2020). https://doi.org/10.1111/cgf.13946

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: Data-Driven Responsive Control of Physics-Based Characters. *ACM Trans. Graph.* 38, 6, Article 206 (Nov. 2019), 11 pages. https://doi.org/10.1145/3355089.3356536

Matthew Brand. 1999. Voice Puppetry. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 21–28. https://doi.org/10.1145/311535.311537

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *CoRR* abs/2005.14165 (2020). arXiv:2005.14165 https://arxiv.org/abs/2005.14165

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-End Object Detection with Transformers. https://doi.org/10.48550/ARXIV.2005.12872

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. *CoRR* abs/2107.03374 (2021). arXiv:2107.03374 https://arxiv.org/abs/2107.03374

Alexander Clegg, Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. 2018. Learning to Dress: Synthesizing Human Dressing Motion via Deep Reinforcement Learning. *ACM Trans. Graph.* 37, 6, Article 179 (dec 2018), 10 pages. https://doi.org/10.1145/3272127.3275048

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. https://doi.org/10.48550/ARXIV.2110.14168

Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. 2009. Robust Task-based Control Policies for Physics-based Characters. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 28, 5 (2009), Article 170.

Marco da Silva, Yeuhi Abe, and Jovan Popović. 2008. Simulation of Human Motion Data using Short-Horizon Model-Predictive Control. *Computer Graphics Forum* 27 (2008).

Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. 2010. Feature-Based Locomotion Controllers. *ACM Transactions on Graphics* 29, 3 (2010).

Salil Deena and Aphrodite Galata. 2009. Speech-Driven Facial Animation Using a Shared Gaussian Process Latent Variable Model. In *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part I* (Las Vegas, Nevada) *(ISVC '09)*. Springer-Verlag, Berlin, Heidelberg, 89–100. https://doi.org/10.1007/978-3-642-10331-5_9

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/ARXIV.1810.04805

Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. 2018. Divide-and-Conquer Reinforcement Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJwelMbR-

F. Sebastin Grassia. 1998. Practical Parameterization of Rotations Using the Exponential Map. *J. Graph. Tools* 3, 3 (March 1998), 29–48. https://doi.org/10.1080/10867651.1998.10487493

Jonathan Ho and Stefano Ermon. 2016. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/cc7e2b878868cbae992d1fb743995d8f-Paper.pdf

Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. 1995. Animating Human Athletics. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 71–78. https://doi.org/10.1145/218380.218414

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-Functioned Neural Networks for Character Control. *ACM Trans. Graph.* 36, 4, Article 42 (jul 2017), 13 pages. https://doi.org/10.1145/3072959.3073663

Daniel Holden, Jun Saito, and Taku Komura. 2016. A Deep Learning Framework for Character Motion Synthesis and Editing. *ACM Trans. Graph.* 35, 4, Article 138 (jul 2016), 11 pages. https://doi.org/10.1145/2897824.2925975

Pengyu Hong, Zhen Wen, and T.S. Huang. 2002. Real-time speech-driven face animation with expressions using neural networks. *IEEE Transactions on Neural Networks* 13, 4 (2002), 916–927. https://doi.org/10.1109/TNN.2002.1021892

Tero Karras, Timo Aila, Samuli Laine, Antti Herva, and Jaakko Lehtinen. 2017. Audio-Driven Facial Animation by Joint End-to-End Learning of Pose and Emotion. *ACM Trans. Graph.* 36, 4, Article 94 (jul 2017), 12 pages. https://doi.org/10.1145/3072959.3073658

Kyungho Lee, Sehee Min, Sunmin Lee, and Jehee Lee. 2021b. Learning Time-Critical Responses for Interactive Character Control. *ACM Trans. Graph.* 40, 4, Article 147 (jul 2021), 11 pages. https://doi.org/10.1145/3450626.3459826

Seyoung Lee, Sunmin Lee, Yongwoo Lee, and Jehee Lee. 2021a. Learning a family of motor skills from a single motion clip. *ACM Trans. Graph.* 40, 4, Article 93 (2021).

Yoonsang Lee, Sungeun Kim, and Jehee Lee. 2010a. Data-Driven Biped Control. *ACM Trans. Graph.* 29, 4, Article 129 (July 2010), 8 pages. https://doi.org/10.1145/1778765.1781155

Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. 2010b. Motion Fields for Interactive Character Locomotion. *ACM Trans. Graph.* 29, 6, Article 138 (dec 2010), 8 pages. https://doi.org/10.1145/1882261.1866160

S. Levine, C. Theobalt, and V. Koltun. 2009. Real-Time Prosody-Driven Synthesis of Body Language. *ACM Transactions on Graphics* 28 (12 2009), 1–10. https://doi.org/10.1145/1618452.1618518

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. 2022. Competition-Level Code Generation with AlphaCode. https://doi.org/10.48550/ARXIV.2203.07814

Angela S. Lin, Lemeng Wu, and Qixing Huang Raymond J. Mooney Rodolfo Corona, Kevin Tai. 2018. Generating Animated Videos of Human Activities from Natural Language Descriptions. In *Proceedings of the Visually Grounded Interaction and Language Workshop at NeurIPS 2018*. http://www.cs.utexas.edu/users/ai-labpub-view.php?PubID=127730

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel van de Panne. 2020. Character Controllers Using Motion VAEs. *ACM Trans. Graph.* 39, 4 (2020).

Libin Liu and Jessica Hodgins. August 2018. Learning Basketball Dribbling Skills Using Trajectory Optimization and Deep Reinforcement Learning. *ACM Transactions on Graphics* 37, 4 (August 2018).

Libin Liu, Michiel van de Panne, and KangKang Yin. 2016. Guided Learning of Control Graphs for Physics-Based Characters. *ACM Transactions on Graphics* 35, 3 (2016).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://doi.org/10.48550/ARXIV.1907.11692

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. *CoRR* abs/2108.10470 (2021). arXiv:2108.10470 https://arxiv.org/abs/2108.10470

Fabian Mentzer, Eirikur Agustsson, Johannes Ballé, David Minnen, Nick Johnston, and George Toderici. 2021. Neural Video Compression using GANs for Detail Synthesis and Propagation. https://doi.org/10.48550/ARXIV.2107.12038

Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. 2017. Learning human behaviors from motion capture by adversarial imitation. *CoRR* abs/1707.02201 (2017). arXiv:1707.02201 http://arxiv.org/abs/1707.02201

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of Complex Behaviors through Contact-Invariant Optimization. *ACM Trans. Graph.* 31, 4, Article

[1] https://actorcore.reallusion.com/

43 (jul 2012), 8 pages. https://doi.org/10.1145/2185520.2185539

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. 2021. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. https://doi.org/10.48550/ARXIV.2112.10741

Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and Jehee Lee. 2019. Learning Predict-and-Simulate Policies from Unorganized Human Motion Data. *ACM Trans. Graph.* 38, 6, Article 205 (Nov. 2019), 11 pages. https://doi.org/10.1145/3355089.3356501

Catherine Pelachaud, Norman Badler, and Mark Steedman. 1996. Generating Facial Expressions for Speech. *Cognitive Science* 20 (03 1996), 1–46. https://doi.org/10.1016/S0364-0213(99)80001-9

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018a. DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills. *ACM Trans. Graph.* 37, 4, Article 143 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201311

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. 2022. ASE: Large-scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters. *ACM Trans. Graph.* 41, 4, Article 94 (July 2022).

Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. 2018b. SFV: Reinforcement Learning of Physical Skills from Videos. *ACM Trans. Graph.* 37, 6, Article 178 (Nov. 2018), 14 pages.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control. *ACM Trans. Graph.* 40, 4, Article 1 (July 2021), 15 pages. https://doi.org/10.1145/3450626.3459670

Matthias Plappert, Christian Mandery, and Tamim Asfour. 2017. Learning a bidirectional mapping between human whole-body motion and natural language using deep recurrent neural networks. *CoRR* abs/1705.06400 (2017). arXiv:1705.06400 http://arxiv.org/abs/1705.06400

Nancy Pollard, Jessica Hodgins, Marcia Riley, and Christopher Atkeson. 2002. Adapting Human Motion for the Control of a Humanoid Robot. 2 (04 2002). https://doi.org/10.1109/ROBOT.2002.1014737

Abhinanda R. Punnakkal, Arjun Chandrasekaran, Nikos Athanasiou, Alejandra Quiros-Ramirez, and Michael J. Black. 2021. BABEL: Bodies, Action and Behavior with English Labels. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. 722–731.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. *CoRR* abs/2103.00020 (2021). arXiv:2103.00020 https://arxiv.org/abs/2103.00020

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. https://doi.org/10.48550/ARXIV.1910.10683

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

Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *CoRR* abs/1706.05098 (2017). arXiv:1706.05098 http://arxiv.org/abs/1706.05098

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. 2015. Policy Distillation. https://doi.org/10.48550/ARXIV.1511.06295

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347 http://arxiv.org/abs/1707.06347

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural State Machine for Character-Scene Interactions. *ACM Trans. Graph.* 38, 6, Article 209 (nov 2019), 14 pages. https://doi.org/10.1145/3355089.3356505

Fuwen Tan, Song Feng, and Vicente Ordonez. 2018. Text2Scene: Generating Abstract Scenes from Textual Descriptions. *CoRR* abs/1809.01110 (2018). arXiv:1809.01110 http://arxiv.org/abs/1809.01110

Jie Tan, Yuting Gu, C. Karen Liu, and Greg Turk. 2014. Learning Bicycle Stunts. *ACM Trans. Graph.* 33, 4, Article 50 (July 2014), 12 pages. https://doi.org/10.1145/2601097.2601121

Guy Tevet, Brian Gordon, Amir Hertz, Amit H. Bermano, and Daniel Cohen-Or. 2022. MotionCLIP: Exposing Human Motion Generation to CLIP Space. https://doi.org/10.48550/ARXIV.2203.08063

Adrien Treuille, Yongjoon Lee, and Zoran Popović. 2007. Near-Optimal Character Animation with Continuous Control. In *ACM SIGGRAPH 2007 Papers* (San Diego, California) *(SIGGRAPH '07)*. Association for Computing Machinery, New York, NY, USA, 7–es. https://doi.org/10.1145/1275808.1276386

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. 2009. Optimizing Walking Controllers. In *ACM SIGGRAPH Asia 2009 Papers* (Yokohama, Japan) *(SIGGRAPH Asia '09)*. Association for Computing Machinery, New York, NY, USA, Article 168, 8 pages. https://doi.org/10.1145/1661412.1618514

Jack M. Wang, Samuel R. Hamner, Scott L. Delp, and Vladlen Koltun. 2012. Optimizing Locomotion Controllers Using Biologically-Based Actuators and Objectives. *ACM Trans. Graph.* 31, 4, Article 25 (jul 2012), 11 pages. https://doi.org/10.1145/2185520.2185521

Tingwu Wang, Yunrong Guo, Maria Shugrina, and Sanja Fidler. 2020. UniCon: Universal Neural Controller For Physics-based Character Motion. arXiv:2011.15119 [cs.GR]

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A Scalable Approach to Control Diverse Behaviors for Physically Simulated Characters. *ACM Trans. Graph.* 39, 4, Article 33 (jul 2020), 12 pages. https://doi.org/10.1145/3386569.3392381

Katsu Yamane, Stuart O. Anderson, and Jessica K. Hodgins. 2010. Controlling humanoid robots with human motion data: Experimental validation. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*. 504–510. https://doi.org/10.1109/ICHR.2010.5686312

Ri Yu, Hwangpil Park, and Jehee Lee. 2021. Human Dynamics from Monocular Video with Dynamic Camera Movements. *ACM Trans. Graph.* 40, 6, Article 208 (dec 2021), 14 pages. https://doi.org/10.1145/3478513.3480504

Y. Yuan, S. Wei, T. Simon, K. Kitani, and J. Saragih. 2021. SimPoE: Simulated Character Control for 3D Human Pose Estimation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 7155–7165. https://doi.org/10.1109/CVPR46437.2021.00708

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. https://doi.org/10.48550/ARXIV.1808.05326

Yunbo Zhang, Wenhao Yu, C. Karen Liu, Charlie Kemp, and Greg Turk. 2020. Learning to Manipulate Amorphous Materials. *ACM Trans. Graph.* 39, 6, Article 189 (nov 2020), 11 pages. https://doi.org/10.1145/3414685.3417868

A STATE AND ACTION REPRESENTATION

We evaluate the effectiveness of our framework by training language-directed control policies for a 3D simulated humanoid character. The character is equipped with a sword and shield, similar to the one used by Peng et al. [2022], with a total of 37 degrees-of-freedom. The character's state $\mathbf{s}_t$ is represented by a collection of features that describes the configuration of the character's body. The features include:

- Height of the root from the ground.
- Rotation of the root in the character's local coordinate frame.
- Local rotation of each joint.
- Local velocity of each joint.
- Positions of the hands, feet, sword and shield in the character's local coordinate frame.

The root is designated to be the pelvis. The character's local coordinate frame is defined with the origin located at the character's pelvis, and the x-axis aligned along the root link's facing direction, with the y-axis aligned with the global up vector. The rotation of each joint is encoded using two 3D vectors, which represent the tangent and normal of the link's local coordinate frame expressed in the link's parent coordinate frame [Peng et al. 2021]. Each action $\mathbf{a}_t$ specifies target rotations for PD controllers positioned at each joint. Following Peng et al. [2021], the target rotations for 3D joints are specified using a 3D exponential map Grassia [1998].

B TASK DETAILS

B.1 Facing Task

The facing task reward is given by:

$$r_t^{\text{task}} = \min\left(\mathbf{d}_t \cdot \mathbf{d}_t^*, 0.5\right) \tag{12}$$

where $\mathbf{d}_t$ is the agent's facing direction. We threshold the reward, which creates an optimal "cone" where the task reward is saturated, allowing the agent to deviate slightly from the target heading in order to better imitate skills.

B.2 Location Task

The location task reward is calculated according to:

$$r_t^{\text{task}} = \begin{cases} 0.2r_t^{\text{pos}} + 0.8r_t^{\text{vel}} & ||\mathbf{x}^* - \mathbf{x}||_2 > \delta_{\text{pos}} \\ 0.8 & ||\mathbf{x}^* - \mathbf{x}||_2 \leq \delta_{\text{pos}} \end{cases} \tag{13}$$

where $\mathbf{x}$ denotes the position of the character's root, and r_t^{pos} encourages the character to be close to the target:

$$r_t^{\text{pos}} = \exp\left(-0.25||\mathbf{x}^* - \mathbf{x}||_2^2\right), \tag{14}$$

r_t^{vel} encourages the character to move towards the target. This velocity reward incentivizes the agent to travel speed of at least $\delta_{\text{vel}} = 0.5$ m/s in the direction of the target, and not travel in any other direction:

$$r_t^{\text{vel}} = \exp\left(-0.25\left(\max(\delta_{\text{vel}} - v_t^{\text{proj}}, 0) + 0.1v_t^{\text{perp}}\right)\right) \tag{15}$$

where

$$v_t^{\text{proj}} = ||\text{proj}_{\mathbf{x}^*}(\mathbf{v}_t)||_2 \tag{16}$$

$$v_t^{\text{perp}} = ||\text{perp}_{\mathbf{x}^*}(\mathbf{v}_t)||_2 \tag{17}$$

define the agent's velocity in the direction of and tangent to the target, respectively. We saturate the task reward when the agent gets within $\delta_{\text{pos}} = 2\text{m}$ of the target, and terminate the episode when the block is knocked over to disincentivize the agent simply running into the block.

B.3 Strike Task

Finally, we have a strike task, where the objective is for the character to knock over a target object. The goal $\mathbf{g}_t = (\tilde{\mathbf{x}}_t^*, \dot{\tilde{\mathbf{x}}}_t^*, \tilde{q}_t^*, \dot{\tilde{q}}_t^*)$ records the target object's position $\tilde{\mathbf{x}}_t^*$, rotation $\tilde{q}_t^*$, linear velocity $\dot{\tilde{\mathbf{x}}}_t^*$, and angular velocity $\dot{\tilde{q}}_t^*$. All features are expressed in the character's local coordinate frame. The task-reward is then given by:

$$r_t^{\text{task}} = \begin{cases} 0.2r_t^{\text{pos}} + 0.8r_t^{\text{vel}} + 0.8r_t^{\text{knock}} & u_t^* \cdot u^{\text{up}} \geq 0.3 \\ 1.4 & u_t^* \cdot u^{\text{up}} < 0.3 \end{cases} \tag{18}$$

where the knock reward incentivizes the agent to knock over the block:

$$r_t^{\text{knock}} = 1 - u_t^* \cdot u^{\text{up}}. \tag{19}$$

Here, u^{up} is the global up vector, and u_t^* is target object's local up vector expressed in the global coordinate frame. The position reward r_t^{pos} and velocity reward r_t^{vel} are the same as those used for the location task. The task reward saturates when the block has been sufficiently tipped over.

B.4 Adaptive Task Weight Schedule

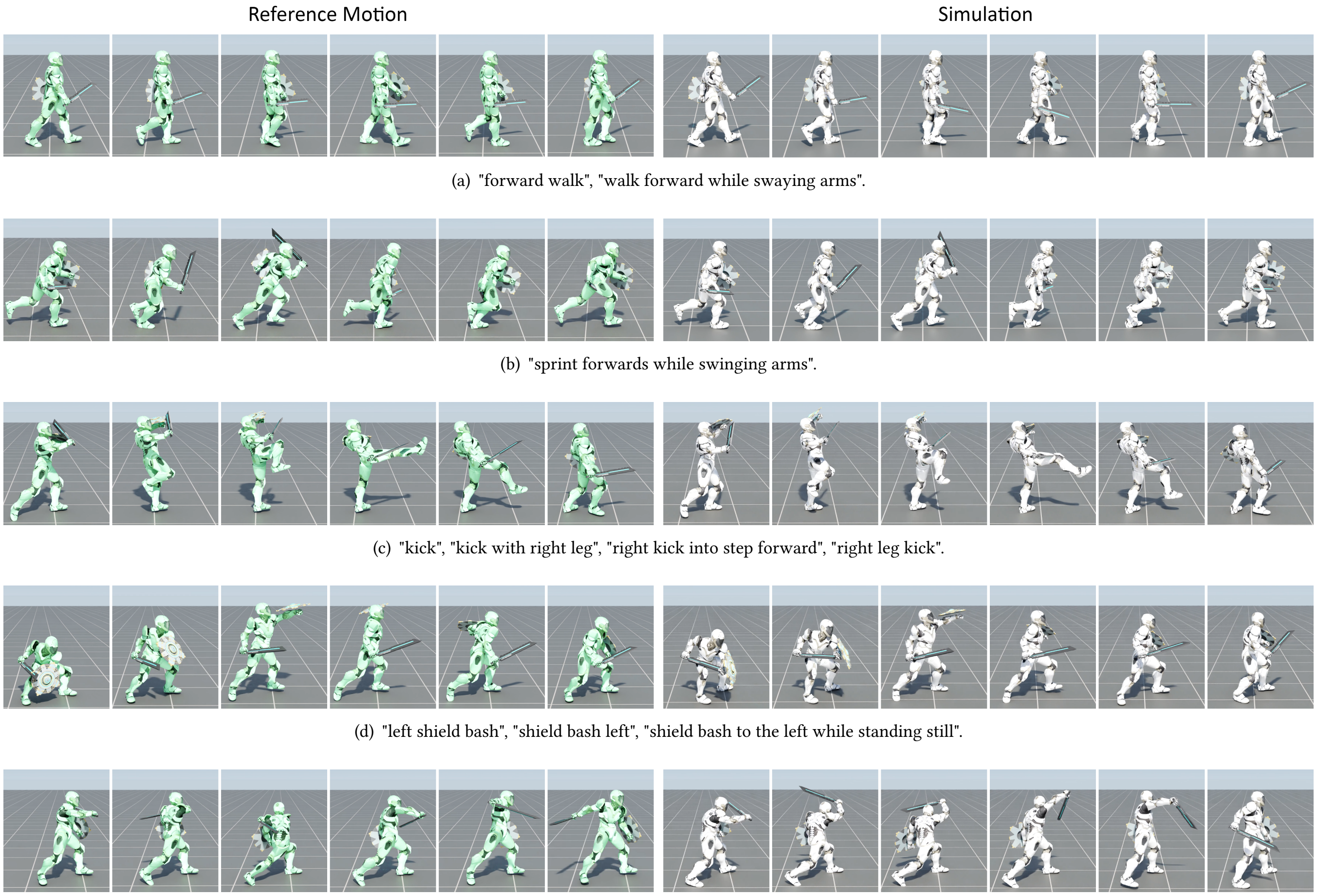
Selecting a weight λ^{task} for the task reward that effectively balances the task and skill reward can be challenging, and can require task-specific tuning. Setting λ^{task} too low can lead to policies that only learn to imitate skills without any regard for the task. Similarly, when λ^{task} is too high, the policy can learn to perform the task using unnatural behaviors, entirely ignoring the skill command. Therefore, instead of using a constant task weight or manually constructing an annealing schedule for λ^{task}, we use a proportional controller to dynamically adjust λ^{task} over the course of the training process, in a similar manner as Mentzer et al. [2021]. The controller is parameterized by a target task reward $\hat{r}^{\text{tar}}$, as well as by a controller gain k_p and a small positive constant ϵ for numerical stability. At epoch i, we calculate the mean task reward $\bar{r}_i^{\text{task}}$ across the experience buffer. We then update the task weight λ_i^{task} according to the error between $\bar{r}_i^{\text{task}}$ and $\hat{r}^{\text{tar}}$ in log-space:

$$\lambda_{i+1}^{\text{task}} = \exp\left(\log\left(\lambda_i^{\text{task}}\right) + k_p\left(\log\left(\hat{r}^{\text{tar}} + \epsilon\right) - \log\left(\bar{r}_i^{\text{task}} + \epsilon\right)\right)\right) \tag{20}$$

The task weight is initialized to be $\lambda_0^{\text{task}} = 3$, and λ_i^{task} is clamped to the range $[0.5, 3]$. For the location task we set a target task reward weight of 0.15, while for the strike task we set a target reward of 0.3. For the facing task we found the controller to be unnecessary and used a constant $\lambda^{\text{task}} = 1$.

C MULTIPLE-CHOICE MODEL EXAMPLE OUTPUTS

In Table 1, we provide examples of task commands and the corresponding object and policy that the multiple-choice QA model



(a) "forward walk", "walk forward while swaying arms".

(b) "sprint forwards while swinging arms".

(c) "kick", "kick with right leg", "right kick into step forward", "right leg kick".

(d) "left shield bash", "shield bash left", "shield bash to the left while standing still".

(e) "slash right", "right swing", "swing sword to the right", "stand still and slash to the right".

Figure 7: Reference motion clips (left side) and their corresponding captions, along with motions produced by a simulated character when directed to perform the reference skills through language commands (right side).

identified. We observe that the QA model is able to correctly identify the user’s intent even when provided with exotic task commands such as “destroy the green guy” or “mosey on down to the maroon saloon”. We also provide several examples where the QA model incorrectly identifies the task and/or object. For example, the model predicts that the task command “go to the blue target” references the strike task instead of the location task, while “go to the blue block” and “go to the blue tower” are correctly identified as the location task. The QA model is also occasionally sensitive to paraphrasing, such as when it correctly identifies the task in “navigate to the lime rectangular prism" but not in “navigate toward the lime rectangular prism".

D COMPARING PADL TO OTHER ADVERSARIAL RL FRAMEWORKS

When training PADL agents (detailed in Section 6), the skill objective explicitly rewards agents for being able to imitate every motion clip in the dataset, using a discriminator trained on the joint distribution of state transitions and skill embeddings. We find that the use of a joint discriminator helps to mitigate mode collapse during PADL training when compared to other work in adversarial reinforcement learning that uses discriminators trained only on the marginal distribution of state transitions. Here we specifically compare our method to two related adversarial RL frameworks, AMP [Peng et al. 2021] and ASE [Peng et al. 2022].

D.1 Comparison to AMP

AMP, like PADL, trains agents using a combination of task and skill rewards. However, since AMP’s skill reward uses a marginal discriminator, mode collapse can occur, where agents focus on imitating a specific subset of skills in the reference motion data while completing the high-level task. PADL’s use of a joint discriminator in the skill reward, where policies are explicitly trained to accomplish the high-level task using different reference skills, can improves a policy’s coverage of the dataset. Moreover, PADL agents, unlike AMP agents, are conditioned on a latent variable encoding the skill to be used. This allows a user to control in real-time which

Table 1: Example task commands and the corresponding object and task identified by the multiple-choice QA model.

Task Command	Identified Object	Identified Task
"knock over the blue block"	"the blue object nearby." ✓	"knock over a specified object." ✓
"knock over the green block"	"the green object nearby." ✓	"knock over a specified object." ✓
"go to the red block"	"the red object nearby." ✓	"navigate to a specified destination."✓
"go to the orange block"	"the orange object nearby."✓	"navigate to a specified destination."✓
"face the purple block"	"the purple object nearby." ✓	"orient himself to face a specified heading."✓
"knock over the purple target"	"the purple object nearby." ✓	"knock over a specified object." ✓
"turn towards the blue target"	"the blue object nearby." ✓	"orient himself to face a specified heading."✓
"turn towards the orange target"	"the orange object nearby." ✓	"orient himself to face a specified heading."✓
"face the orange target"	"the orange object nearby." ✓	"orient himself to face a specified heading."✓
"face the purple target"	"the purple object nearby." ✓	"orient himself to face a specified heading."✓
"go to the blue target"	"the blue object nearby." ✓	"knock over a specified object." ✗
"topple the red tower"	"the red object nearby."✓	"knock over a specified object." ✓
"face the orange obelisk"	"the orange object nearby." ✓	"orient himself to face a specified heading." ✓
"navigate to the lime rectangular prism"	"the green object nearby." ✓	"navigate to a specified destination." ✓
"navigate toward the lime rectangular prism"	"the green object nearby." ✓	"orient himself to face a specified heading." ✗
"look at the stop sign"	"the red object nearby." ✓	"orient himself to face a specified heading." ✓
"watch the sunset"	"the red object nearby." ✓	"orient himself to face a specified heading." ✓
"knock over the cobalt block"	"the red object nearby." ✗	"knock over a specified object." ✓
"get close to the violet marker"	"the purple object nearby."✓	"orient himself to face a specified heading." ✗
"destroy the green guy"	"the green object nearby." ✓	"knock over a specified object." ✓
"mosey on down to the maroon saloon"	"the red object nearby."✓	"navigate to a specified destination." ✓

skills a trained agent uses to accomplish a task, which is crucial for effective language control.

D.2 Comparison to ASE

Both ASE low-level controllers and PADL controllers are conditioned on skill latents, allowing the skill the agent uses to be dynamically controlled. During ASE training, latents are drawn randomly from a prior distribution (e.g. the unit sphere); the policy learns a meaningful representation on this latent space throughout training using a marginal discriminator combined with an encoder that promotes high mutual information between a latent and its corresponding policy trajectory. This approach too can lead to mode collapse, with only a subset of skills from the reference dataset being represented in the latent space. PADL mitigates this type of mode collapse by assigning a distinct motion latent to every motion clip in the reference dataset (these latents are learned in the Skill Embedding stage), guaranteeing that every motion clip is represented in the latent space.

ALGORITHM 1: Multi-Task Aggregation

```
agentState ← agent state;
while not done do
    skillLatent = $\text{Enc}_l$(getSkillCommand());
    policyIdx, objectIdx = QAModel(getTaskCommand());
    policy = policies[policyIdx];
    targetObjectState = objects[objectIdx];
    action = policy(agentState, skillLatent, targetObjectState);
    agentState = env.step(action);
end
```

In one of our early experiments developing language-controlled animation systems, we attached a language head on top of an ASE low-level controller. We created a dataset of (latent, caption) pairs by sampling latents from the unit sphere, recording trajectories from a pre-trained controller checkpoint with those latents, and annotating the trajectories with natural language. We then trained a small MLP to reverse the annotation process and map the BERT embeddings of a trajectory's caption to the corresponding latent that produced the trajectory. This approach allowed for a policy's skill to be controlled with language, but is annotation inefficient, since each dataset of (latent, caption) pairs is only applicable for a specific checkpoint's learned latent space. A different ASE checkpoint (which possesses a different learned latent space) requires the collection of an entirely new dataset of annotations. Moreover, due to mode-collapse, more complicated skills in the dataset were often not represented in the policy's latent space.