A. Additional Implementation Details

Similar to Magic123 [27] and 4DFY [2], we first generate the static 3D objects via NeRF representation using joint score distillation from MVDream [34] and Stable Diffusion 2.1 [31]. After obtaining the static objects, we convert them to point clouds which are consecutively used to initialize 3D Gaussians. Our full model utilizes point clouds containing 60,000 colored points. We preprocess 60 nearest neighbors for each 3D Gaussian in order to speed up the calculation of $\mathcal{L}_{\text{rigidity}}$. In the dynamic optimization stage, we randomly assign training iterations to adopt single-object rendering (with a probability of 0.2) or compositional rendering (with a probability of 0.8). We train for 3,000 iterations in the dynamic stage with a learning rate of 1e-4 for the deformation MLP. In each iteration, we render 16 frames via uniformly sampled timesteps. We use the frozen video diffusion model, Zeroscope [1], in our experiments. To improve the 2D appearance, we also randomly sample 4 frames out of 16 rendered frames for image score distillation, where the Stable Diffusion 2.1 [31] is used as the image diffusion model. In our experiments, we compare with two open-source baseline text-to-4D generation methods, 4DFY [2] and Animate124 [55].

Our overall loss function can be summarized as follows,

$$\mathcal{L} = \mathcal{L}_{\text{SDS-static}} + \mathcal{L}_{\text{SDS-dynamic}} + \mathcal{L}_{\text{reg}}, \tag{11}$$

where $\mathcal{L}_{\text{reg}}$ refers to regularizations:

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{contact}} + \omega_1 \mathcal{L}_{\text{acc}} + \omega_2 \mathcal{L}_{\text{rigidity}}. \tag{12}$$

ω_1 and ω_2 are weighting coefficients, set to $1e-4$ and $1e3$, respectively. For MVDream [34], we use negative prompt as their default configuration, "ugly, bad anatomy, blurry, pixelated, obscure, unnatural colors, poor lighting, dull, and unclear, cropped, lowres, low quality, artifacts, duplicate, morbid, mutilated, poorly drawn face, deformed, dehydrated, bad proportions". For Zeroscope [1], we use negative prompts that avoid static generations, "static, low motion, static statue, not moving, no motion, text, watermark, copyright, blurry, nsfw".

For baseline methods 4DFY [2] and Animate124 [55], we use the codebase released by the authors. For 4DFY [2], since it is a text-to-4D generation method, we use the same scene text description when generating 4D results. For Animate124 [55], due to its image-to-4D nature, we use the static assets produced by our method to obtain the initial image for Animate124. Specifically, we place the static assets at the $t = 0$ position in the trajectory and render a static image that does not contain occlusions. Then, we follow the workflow of Animate124 first to perform textual inversion and then the following static and dynamic stages.

B. More Evaluation on Trajectory Generation

Semantic Alignment about Global Trajectory. In our pipeline, we leverage a large language model to generate object trajectories. Beyond evaluating the success rate of these trajectories in Sec. 4.2, we further assess their semantic alignment with the corresponding text prompts. This alignment is implicitly captured by human preferences on text alignment (TA) in Tab. 1. At this point, we focus on the global movement (trajectories) between objects. This is non-trivial due to the lack of reliable evaluation metrics or datasets. Fortunately, Large Multimodal Models (LMMs), particularly GPT-4Vision (GPT-4V) [50], have shown remarkable capabilities in reasoning about the three-dimensional physical world. Inspired by humans' ability to perform 3D reasoning tasks using 2D visual information under language guidance, we follow [46] and utilize GPT-4V for our 4D model evaluation task. To simplify the task, we instruct GPT-4V to evaluate the trajectories between two objects. Complex scenes with multiple assets can be reduced to the global displacement between each pair of assets. Specifically, we define the following five categories of relative displacements between two objects:

- One object is static relative to the other (e.g. a bird perching on the back of a buffalo).
- One object approaches the other (e.g. a dog chasing a ball).
- One object moves in the opposite direction to the other (e.g. a rabbit hopping away from a fox).
- One object revolves around the other in a circular (or elliptical) way (e.g. a horse running around a tree).
- One object moves along a random path beside the other (e.g. a butterfly fluttering randomly near a flower).

For each category, we propose 20 distinct scenes and prompt LLM for trajectory design. For each scene, we draw a sequence of images depicting the movement of one object relative to the other in a 3D coordinate space along the designed trajectory, with bounding boxes denoting objects. We uniformly sample 4 images along the trajectory and prompt GPT-4V to classify into one of the five target categories. We show one example in Fig. 5. The experiment shows that overall, 92% (85%-95% in 5 categories) of the designed trajectories are correctly labeled by GPT-4V. It suggests that the designed trajectories align very well with the text prompts.

C. LLM Prompting Visualization

In our framework, we take advantage of an LLM model (ChatGPT-4) to automatically decompose the scene into multiple assets, and design their moving trajectories. We visualize this procedure in Fig. 6 and Fig. 7, where we provide a conversation between a human user and ChatGPT-4 to accomplish this task. The output of ChatGPT includes a function that delineates the moving trajectory of the object. We also visualize the trajectory in Fig. 8. We found these templates to be very robust for trajectory prompting. The trajectory preparation process also incorporates collision checking, optional trajectory truncation, and re-prompting, forming a robust automated pipeline.

For collision checking, to efficiently simulate object occupancy along the trajectory, we approximate each object with a pre-sized rectangular cuboid, as shown in Fig. 5. Despite its simplicity, this strategy works very well in practice for coarse-level spatial planning. This is mainly because the object-level 3DGS, though dynamic in geometry, typically remains bounded within the defined cuboids, thanks to the pre-specification of object scales. To improve reproducibility, we fix the random seed for LLM calls and standardize the prompts with explicit spatial constraints and example formats. Although GPT-4 is inherently probabilistic, this combination of prompt engineering and automated post-processing ensures high reliability. Also, for the scenes that are too complex for ChatGPT to handle, our framework supports human involvement to manually decompose the scene and design the trajectories.

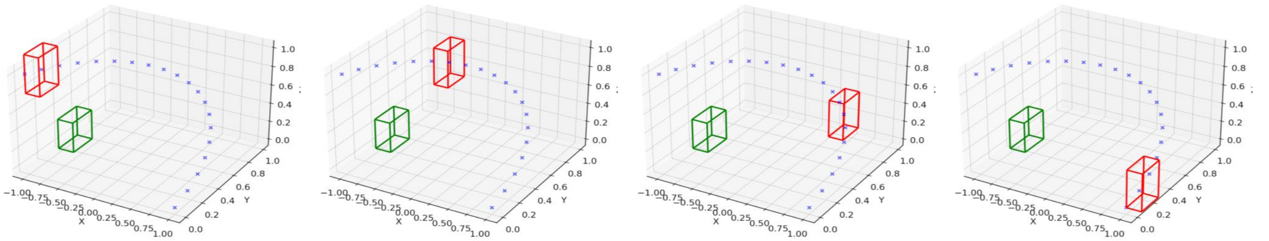


Figure 5. A visualization of objects moving along the trajectory designed by LLM, which is prompted into GPT-4V to justify the relative displacement category (class 4 in this example).

D. Failure Case Analysis

Failure Cases for LLM-based Trajectory Design. Failure cases for LLM trajectory generation primarily stem from *prompt ambiguity* and *scene complexity*. We tested over 50 scenes and found that vague spatial descriptions (e.g., 'a butterfly circling while chasing a fox') reduce success rates by around 20% compared to unambiguous prompts. For complex scenes (over four objects with curved paths), collision-free trajectories often require 4+ LLM trials. Our pipeline mitigates this via automated collision checking and re-prompting, recovering valid trajectories in 85% of cases. These observations also align with our experimental analysis in Tab. 2 in the main paper.

Failure Cases for Scene Generation. While our framework performs robustly across a wide variety of prompts, it may encounter challenges in scenarios that require complex *physical reasoning* or fine-grained *cause-effect understanding*, e.g. prompts involving fluid dynamics (e.g., "a glass of water spilling on the table"). This is because the underlying generative models are limited in understanding physical laws. For future work, we plan to incorporate more *physics-aware* generative models or design trajectory constraints informed by physics engines to enhance realism in such scenes.

E. Prompt Taxonomy and Diversity Analysis

To evaluate the diversity of our text-to-4D generation benchmark, we design a set of 20 prompts covering a wide range of object counts, interaction types, and motion ranges. This collection is used for quantitative metrics (QAlign, CLIP-T) and the user study reported in Tab.1 in the main paper. The taxonomy follows three orthogonal axes:

- **Object count:** *Two objects* (10 prompts), *Three objects* (6 prompts), *Four objects* (4 prompts).
- **Interaction type:** *Independent motion* (objects move without contact), *Loose interaction* (occasional proximity or crossing paths), *Physical contact* (sustained collisions or joint actions).
- **Motion range:** Defined based on the normalized displacement of each object relative to the overall scene scale. *Small* indicates subtle local deformations or displacements less than 10% of the scene size, *Medium* corresponds to 10–30% displacement, and *Large* involves long-range movements covering more than 30% of the scene.

Table 4 enumerates all prompts and their assigned categories. Representative examples include "a baby bunny jumping onto a stack of pancakes" (two objects, physical contact, medium motion) and "four balloons drifting in the wind" (four objects, independent motion, large motion). The number of prompts we design (20) is comparable to prior 4D generation benchmarks, such as 4DFY [2], TC4D [3], and AYG [21]. This taxonomy confirms that our evaluation set covers diverse object counts, interaction complexities, and motion amplitudes, providing a balanced benchmark for multi-object 4D scene generation.

F. Limitations and Future Work

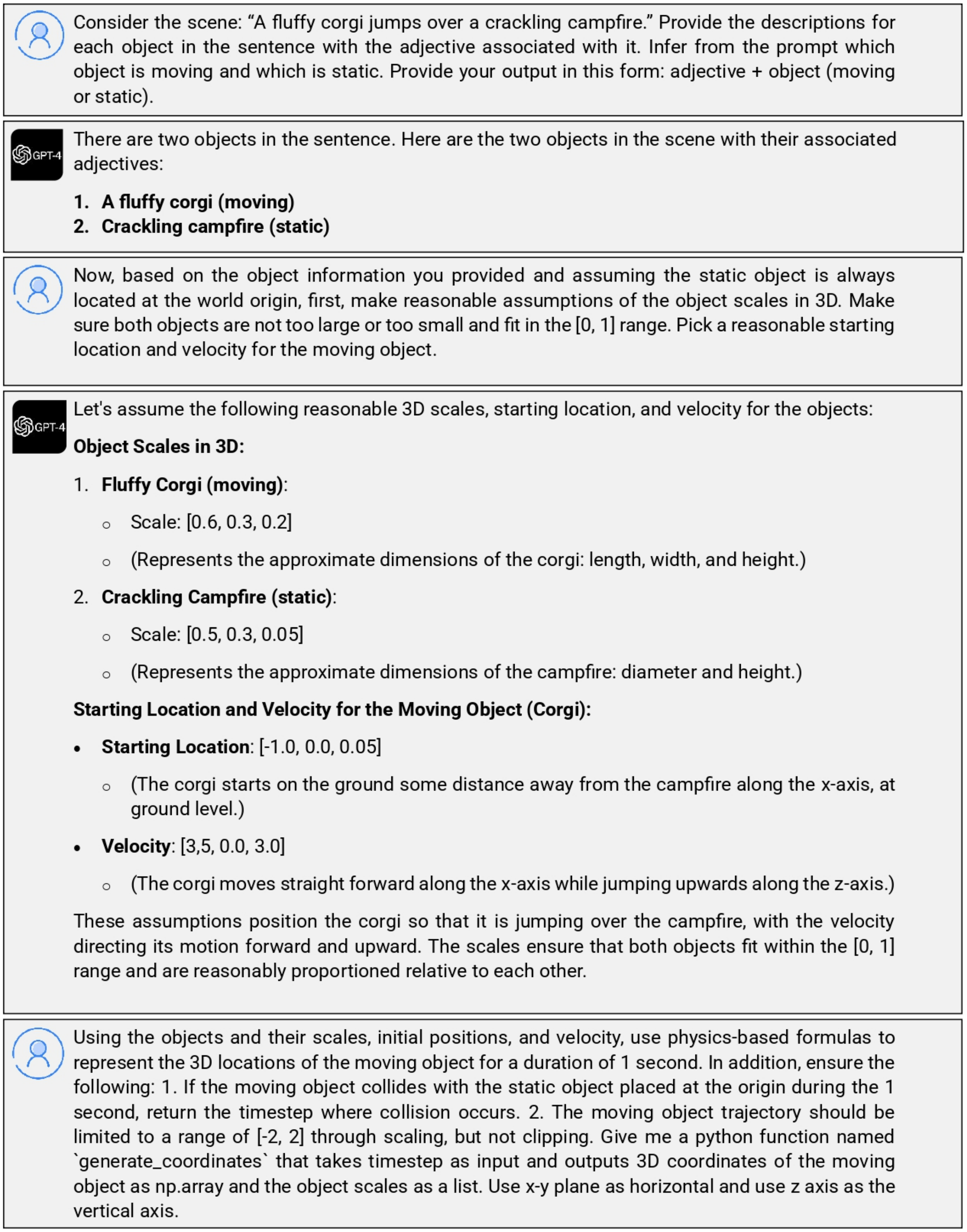
Despite the exciting results produced by Comp4D, our framework still has some limitations. First, generating physically realistic motion remains a significant challenge, as current video models struggle to understand the physical properties of the visual world, leading to artifacts such as unnatural deformations or morphing artifacts. These limitations become more pronounced in complex scenes with multiple objects, where ensuring coherent motion and plausible dynamics is difficult. Second, the temporal consistency and controllability of generated videos remain constrained by existing video generative models. Extending motion duration while maintaining visual coherence and adherence to real-world physics is an open problem, hindering the creation of high-quality, dynamic 4D content. Future work should focus on improving the fidelity of object interactions and enhancing motion realism to support more practical and diverse applications.

G. Broad Impact

Comp4D stands to significantly transform digital content creation by enabling the generation of rich, dynamic 4D scenes that more accurately reflect real-world interactions. By breaking down textual prompts into distinct, movable components and reassembling them with precision, this framework overcomes traditional dataset limitations and object-centric biases. Its innovative integration of diffusion models across text-to-image, text-to-video, and text-to-3D domains not only enhances visual quality and motion fidelity but also paves the way for more immersive experiences in gaming, film, and virtual/augmented reality. In essence, Comp4D could democratize advanced scene synthesis, catalyzing further research and development in multi-dimensional content creation while broadening the creative toolkit available to both professionals and enthusiasts.

Prompt	**Objects**	**Interaction**	**Motion**
a butterfly flies towards a flower	2	loose	medium
a seal balancing a colorful beach ball on its nose	2	physical	small
a fluffy corgi jumping over a cracking campfire	2	loose	medium
a frog jumping onto a lotus leaf	2	physical	small
a parrot jumping onto a book	2	physical	small
a dog running onto a skateboard	2	physical	large
a goat walking towards a cup of beer	2	physical	large
a baby bunny jumping onto a stack of pancakes	2	physical	medium
a colorful parrot flying towards a white parrot	2	loose	large
a fluffy corgi catching a green tennis ball	2	loose	large
two fishes swimming around a rock	3	independent	medium
three penguins waddling	3	independent	small
two juggling clowns tossing pins to each other	3	physical	medium
two drones gliding along a forest trail	3	loose	medium
two friends passing a beach ball	3	physical	large
a cat and a dog playing with a ball of yarn	3	physical	medium
three paper airplanes gliding in a circle over grass	4	loose	medium
four balloons drifting in the wind	4	independent	large
three birds circling above a lighthouse	4	independent	large
a goat and a sheep walking, a seagull flying, around a lamppost	4	loose	medium

Table 4. Complete list of 20 prompts used for evaluation with their taxonomy labels. Object count is given as the number of independently modeled objects. Interaction types are *independent*, *loose* (occasional proximity), or *physical* (sustained contact). Motion range is classified as *small*, *medium*, or *large*.



Consider the scene: “A fluffy corgi jumps over a crackling campfire.” Provide the descriptions for each object in the sentence with the adjective associated with it. Infer from the prompt which object is moving and which is static. Provide your output in this form: adjective + object (moving or static).

There are two objects in the sentence. Here are the two objects in the scene with their associated adjectives:

1. **A fluffy corgi (moving)**
2. **Crackling campfire (static)**

Now, based on the object information you provided and assuming the static object is always located at the world origin, first, make reasonable assumptions of the object scales in 3D. Make sure both objects are not too large or too small and fit in the [0, 1] range. Pick a reasonable starting location and velocity for the moving object.

Let's assume the following reasonable 3D scales, starting location, and velocity for the objects:

Object Scales in 3D:

1. **Fluffy Corgi (moving)**:
   - Scale: [0.6, 0.3, 0.2]
   - (Represents the approximate dimensions of the corgi: length, width, and height.)
2. **Crackling Campfire (static)**:
   - Scale: [0.5, 0.3, 0.05]
   - (Represents the approximate dimensions of the campfire: diameter and height.)

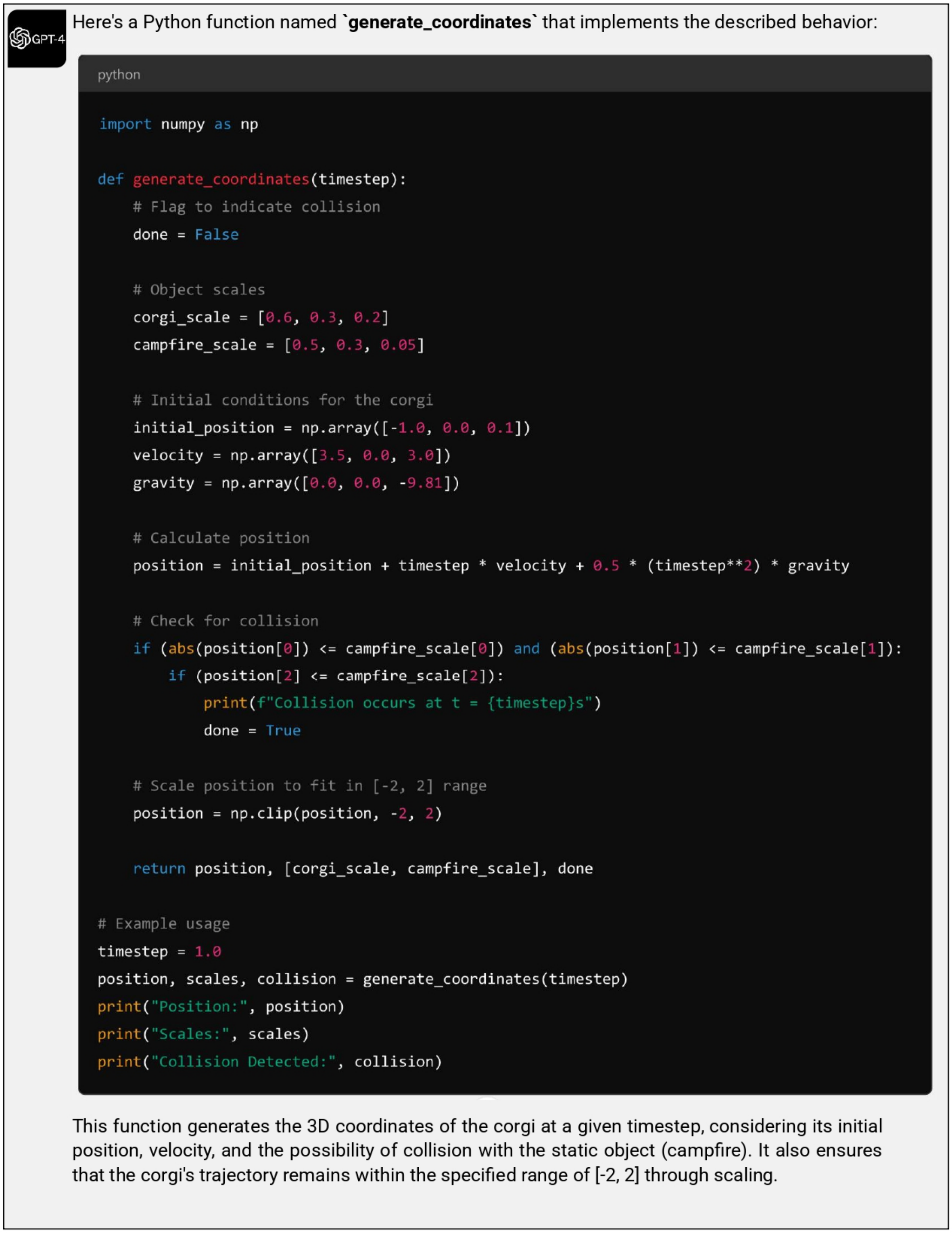
Starting Location and Velocity for the Moving Object (Corgi):

- **Starting Location**: [-1.0, 0.0, 0.05]
  - (The corgi starts on the ground some distance away from the campfire along the x-axis, at ground level.)
- **Velocity**: [3,5, 0.0, 3.0]
  - (The corgi moves straight forward along the x-axis while jumping upwards along the z-axis.)

These assumptions position the corgi so that it is jumping over the campfire, with the velocity directing its motion forward and upward. The scales ensure that both objects fit within the [0, 1] range and are reasonably proportioned relative to each other.

Using the objects and their scales, initial positions, and velocity, use physics-based formulas to represent the 3D locations of the moving object for a duration of 1 second. In addition, ensure the following: 1. If the moving object collides with the static object placed at the origin during the 1 second, return the timestep where collision occurs. 2. The moving object trajectory should be limited to a range of [-2, 2] through scaling, but not clipping. Give me a python function named \`generate_coordinates\` that takes timestep as input and outputs 3D coordinates of the moving object as np.array and the object scales as a list. Use x-y plane as horizontal and use z axis as the vertical axis.

Figure 6. Automatic scene decomposition and trajectory design with ChatGPT-4 (part 1). We provide an example of the conversation between a human user and ChatGPT-4 for scene decomposition and trajectory design. Given a text description, ”A fluffy corgi jumps over a crackling campfire”, the human user first prompts the ChatGPT-4 to extract the component objects from the description, followed by scale assignment over the component objects. Then, the human user prompts the ChatGPT-4 to give initial positions, velocities, and moving trajectories of the objects.

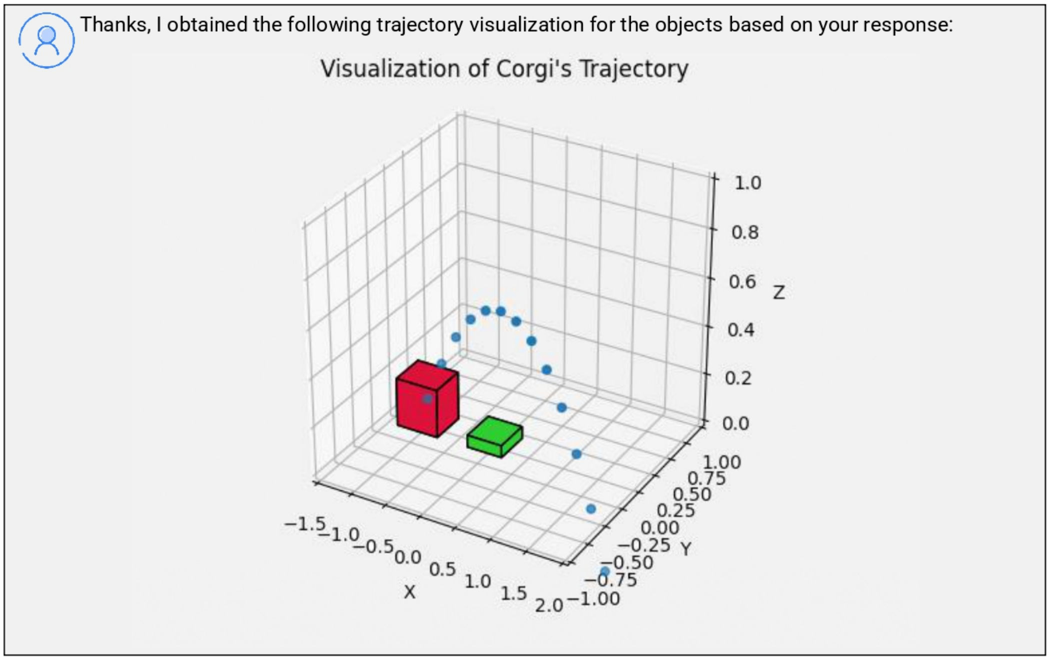


Here's a Python function named **`generate_coordinates`** that implements the described behavior:

```python
import numpy as np

def generate_coordinates(timestep):
    # Flag to indicate collision
    done = False

    # Object scales
    corgi_scale = [0.6, 0.3, 0.2]
    campfire_scale = [0.5, 0.3, 0.05]

    # Initial conditions for the corgi
    initial_position = np.array([-1.0, 0.0, 0.1])
    velocity = np.array([3.5, 0.0, 3.0])
    gravity = np.array([0.0, 0.0, -9.81])

    # Calculate position
    position = initial_position + timestep * velocity + 0.5 * (timestep**2) * gravity

    # Check for collision
    if (abs(position[0]) <= campfire_scale[0]) and (abs(position[1]) <= campfire_scale[1]):
        if (position[2] <= campfire_scale[2]):
            print(f"Collision occurs at t = {timestep}s")
            done = True

    # Scale position to fit in [-2, 2] range
    position = np.clip(position, -2, 2)

    return position, [corgi_scale, campfire_scale], done

# Example usage
timestep = 1.0
position, scales, collision = generate_coordinates(timestep)
print("Position:", position)
print("Scales:", scales)
print("Collision Detected:", collision)
```

This function generates the 3D coordinates of the corgi at a given timestep, considering its initial position, velocity, and the possibility of collision with the static object (campfire). It also ensures that the corgi's trajectory remains within the specified range of [-2, 2] through scaling.

Figure 7. Automatic scene decomposition and trajectory design with ChatGPT-4 (part 2).



Thanks, I obtained the following trajectory visualization for the objects based on your response:

Figure 8. Automatic scene decomposition and trajectory design with ChatGPT-4.