Collaborative Construction and Communication in Minecraft

Julia Hockenmaier
(with Anjali Narayan-Chen and Prashant Jayannavar)
University of Illinois

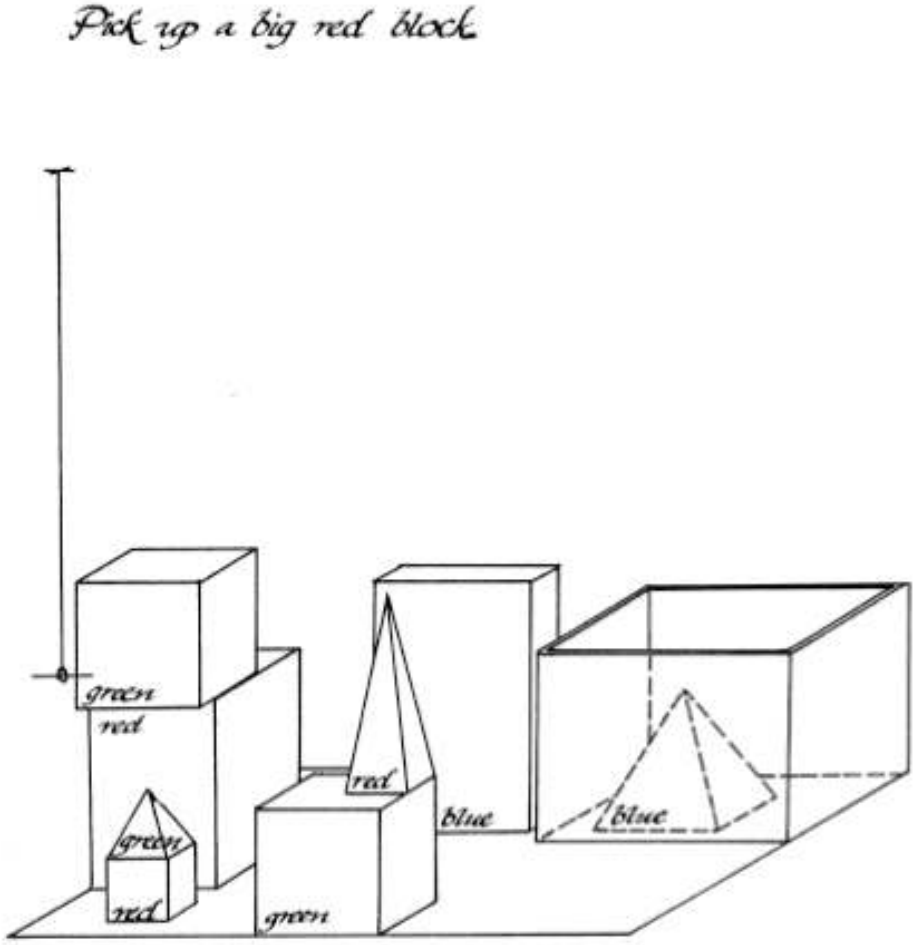
http://juliahmr.cs.illinois.edu/Minecraft

SpLU 2020
Third International Workshop on Spatial Language Understanding

DARPA's Communicating with Computers (CwC) program

CwC aims to enable
symmetric **communication**
between computers and people
in collaborative contexts.

The Blocks World use case:
Humans and machines communicate
to build a given target structure
with toy blocks.

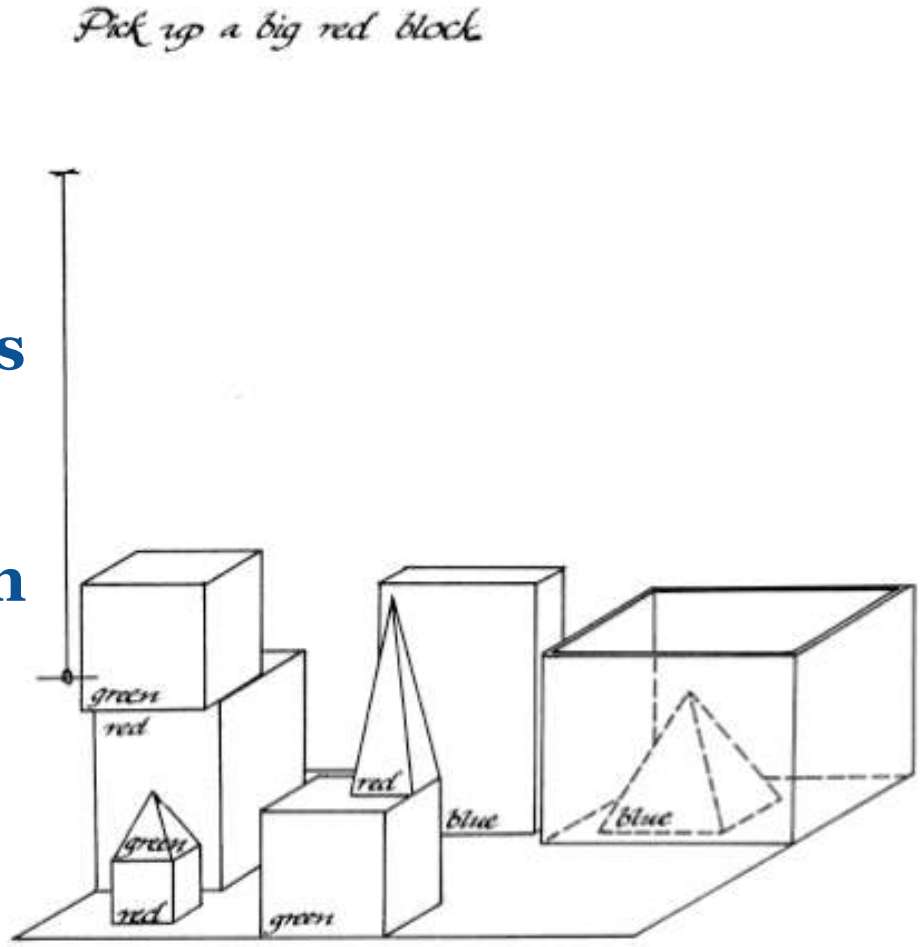


Blocks World: Winograd's SHRDLU (1971)

SHRDLU had a symbolic representation of a scene with several different types of blocks (to simulate an **immobile robot with an arm**)

Users could **instruct SHRDLU to move blocks** in this scene (and ask questions about the scene)

But SHRDLU was based entirely on **handwritten symbolic rules and domain knowledge**.



Can modern systems learn to perform this task without handwritten rules?

Minecraft as a virtual platform for NLP

Popular multi-player gaming platform where **avatars navigate in a 3D world** and **manipulate block-like materials**

Microsoft's **Project Malmo API** makes it possible to use Minecraft for reinforcement learning and other AI research.

We show that this makes Minecraft a great virtual platform to study **interactive, situated language generation & understanding**.

We can use Minecraft to simulate a **Blocks World for embodied agents**

The Minecraft Collaborative Building Task

(Narayan-Chen, Jayannavar, Hockenmaier, ACL 2019)

The Architect

knows the Target **observes the Builder**

The Builder

has to build a copy of the Target

Chat Interface

A: In about the middle
build a column five tall

A: then two more to the left
of the top to make a 7

A: now a yellow 6

A: the long edge of the 6 aligns
with the stem of the 7
and faces right

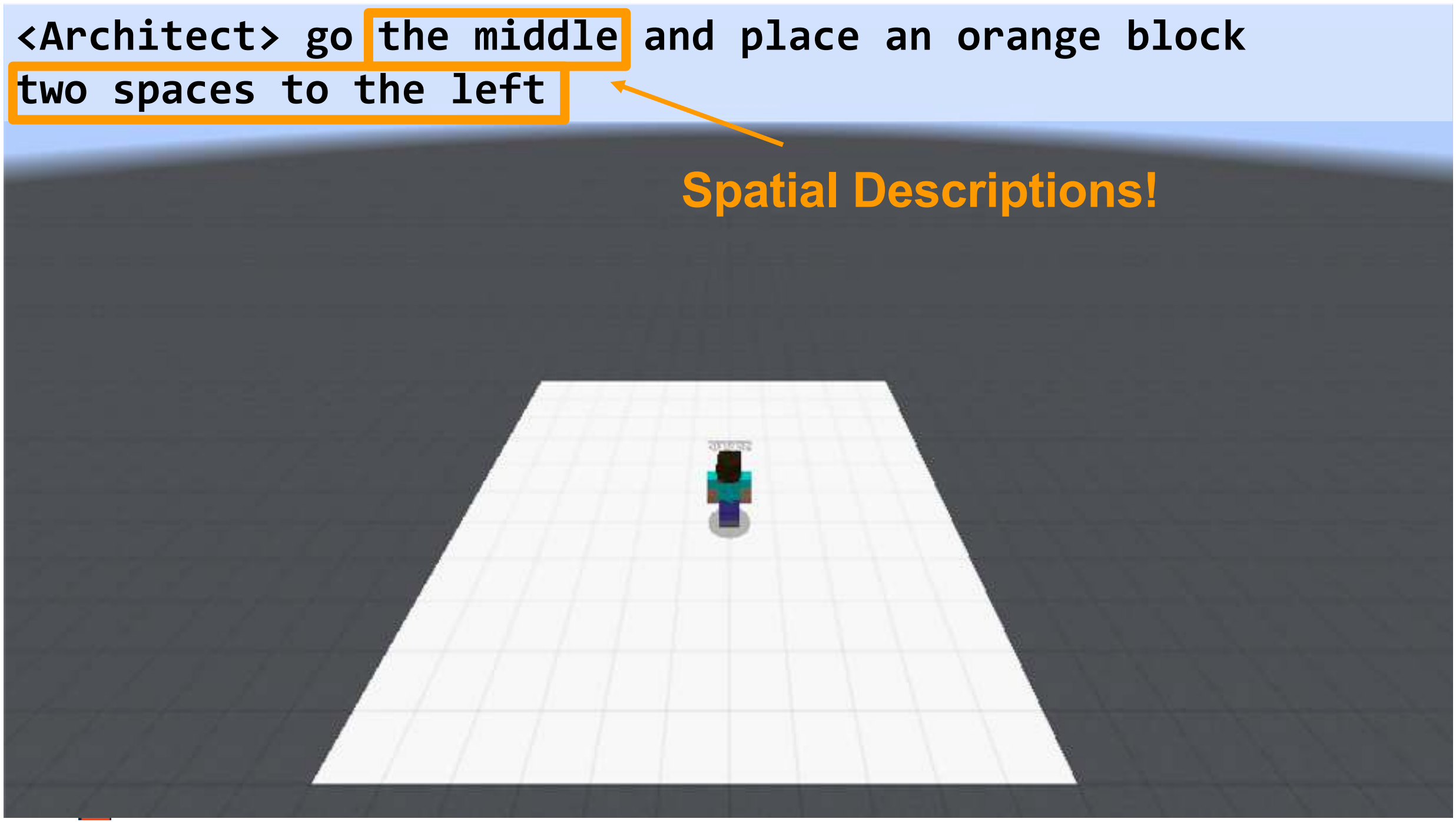
B: where does the 6 start?

A: behind the 7 from your
perspective

How do People Perform This Task?

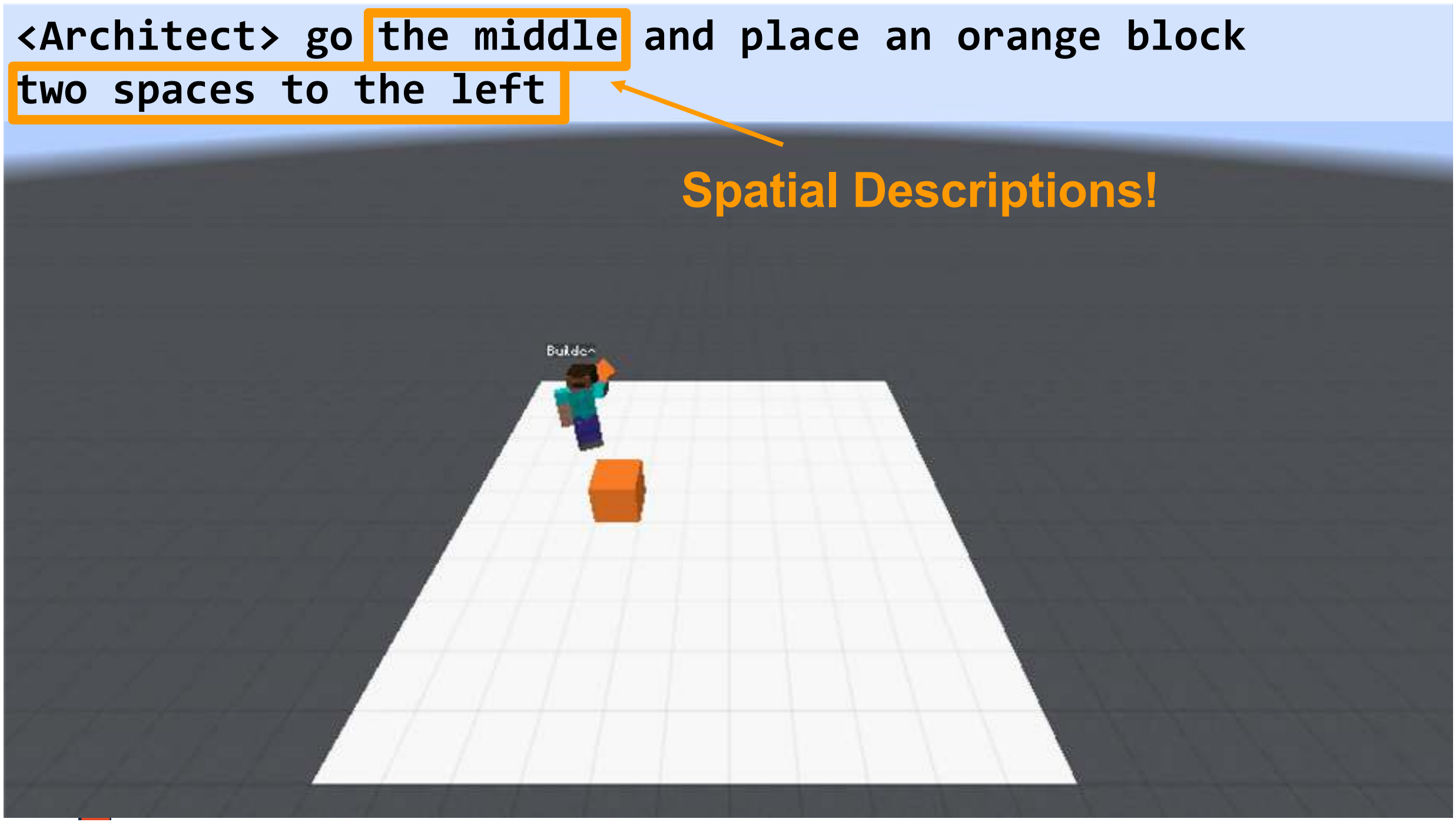
<Architect> go the middle and place an orange block two spaces to the left

Spatial Descriptions!



<Architect> go the middle and place an orange block two spaces to the left

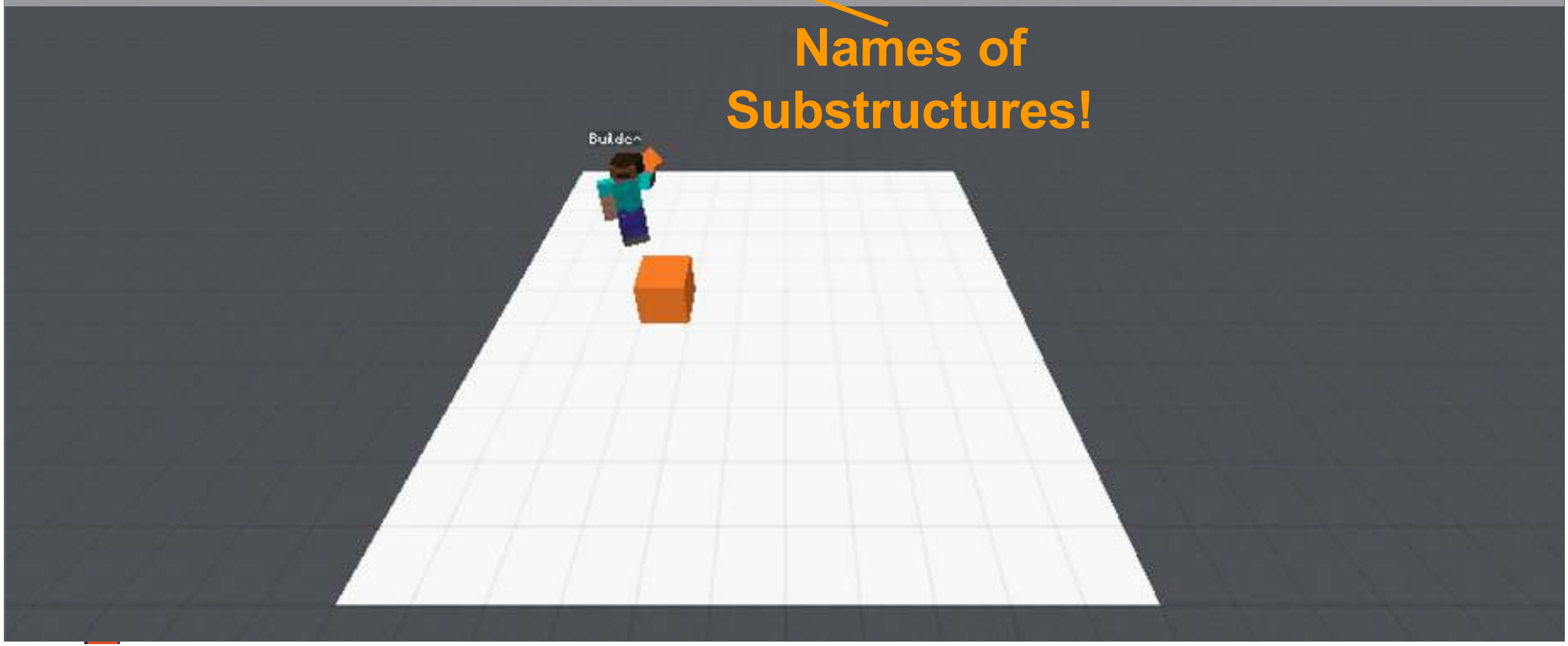
Spatial Descriptions!



<Architect> go the middle and place an orange block two spaces to the left

<Architect> now make a staircase with 2 stairs left and 2 right with orange

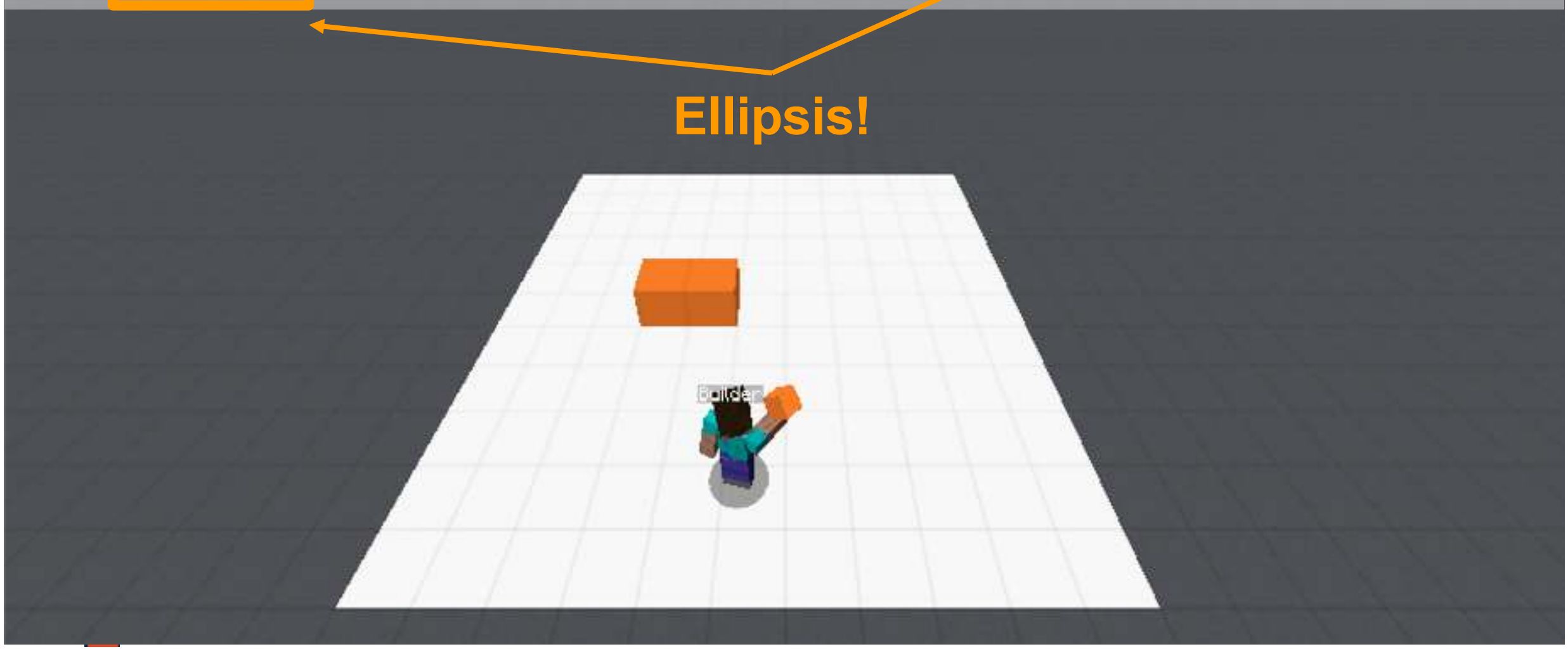
Names of Substructures!



<Architect> go the middle and place an orange block two spaces to the left

<Architect> now make a staircase with 2 stairs left and 2 right with orange

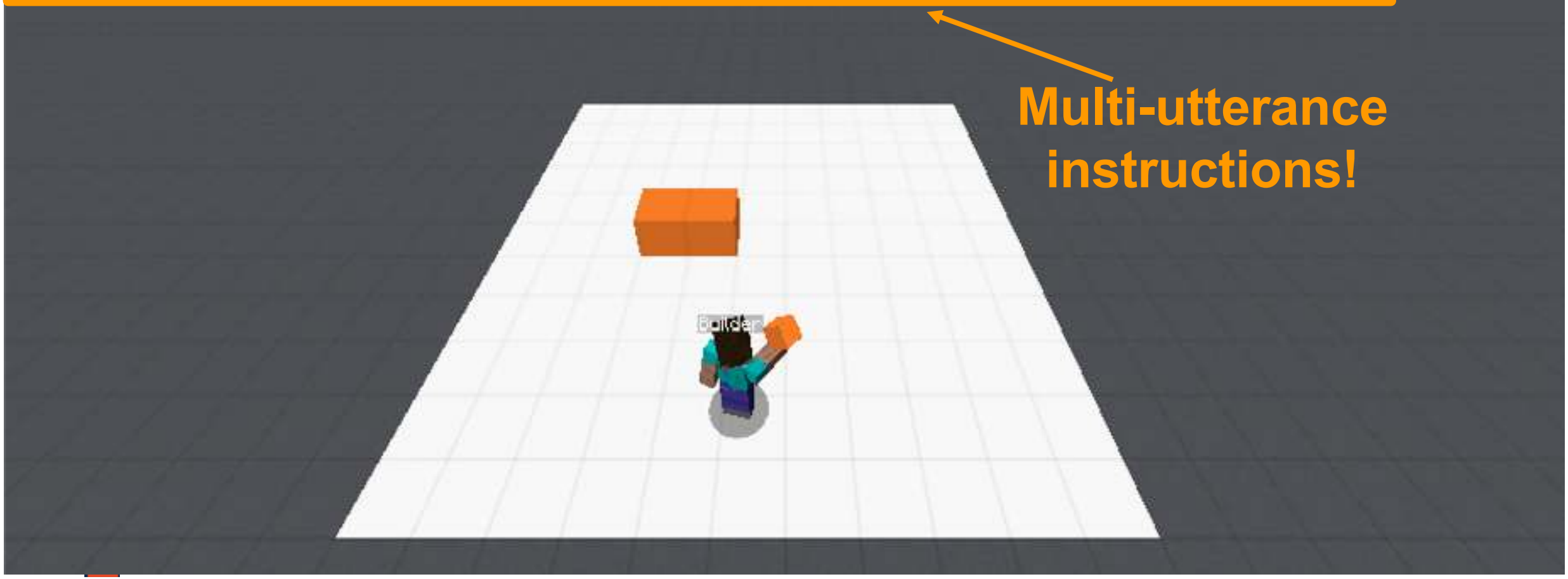
Ellipsis!



<Architect> go the middle and place an orange block two spaces to the left

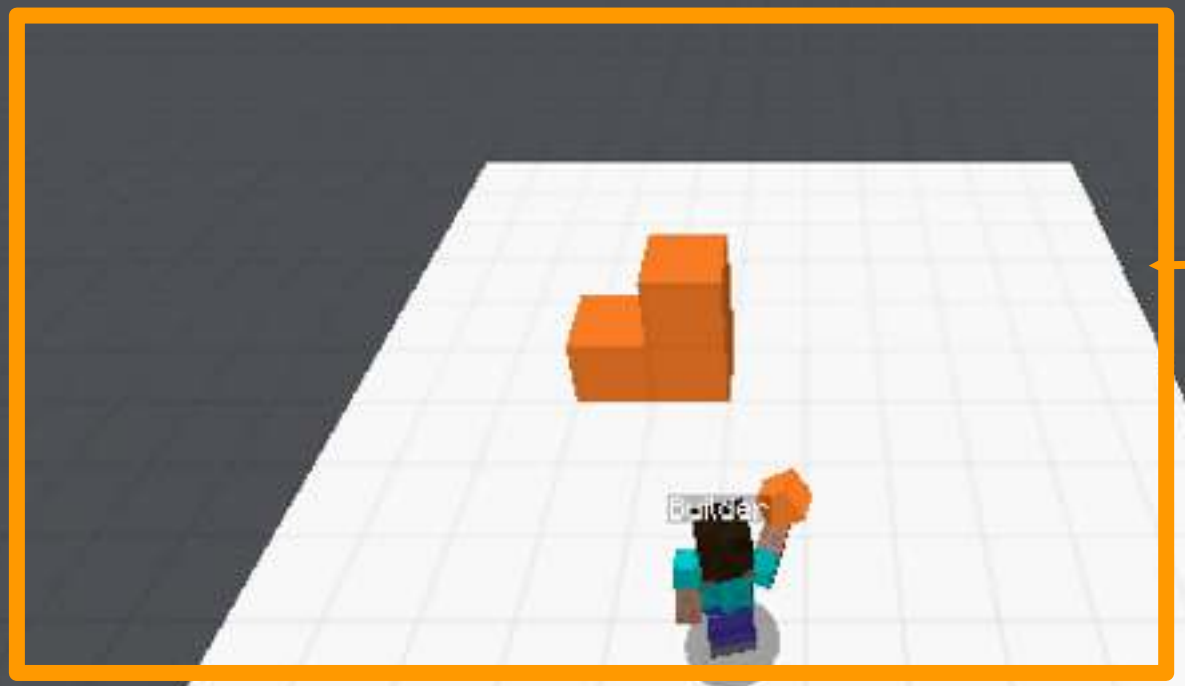
<Architect> now make a staircase with 2 stairs left and 2 right with orange

<Architect> so it will look like a v



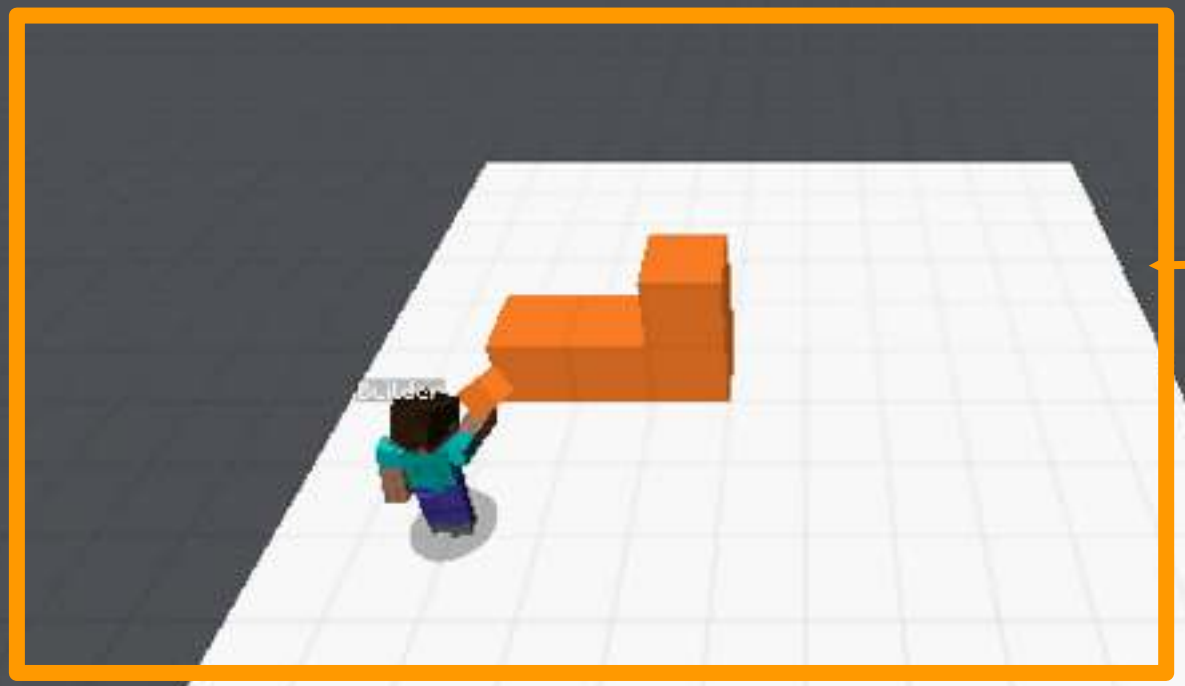
Multi-utterance instructions!

<Architect> go the middle and place an orange block two spaces to the left
<Architect> now make a staircase with 2 stairs left and 2 right with orange
<Architect> so it will look like a v



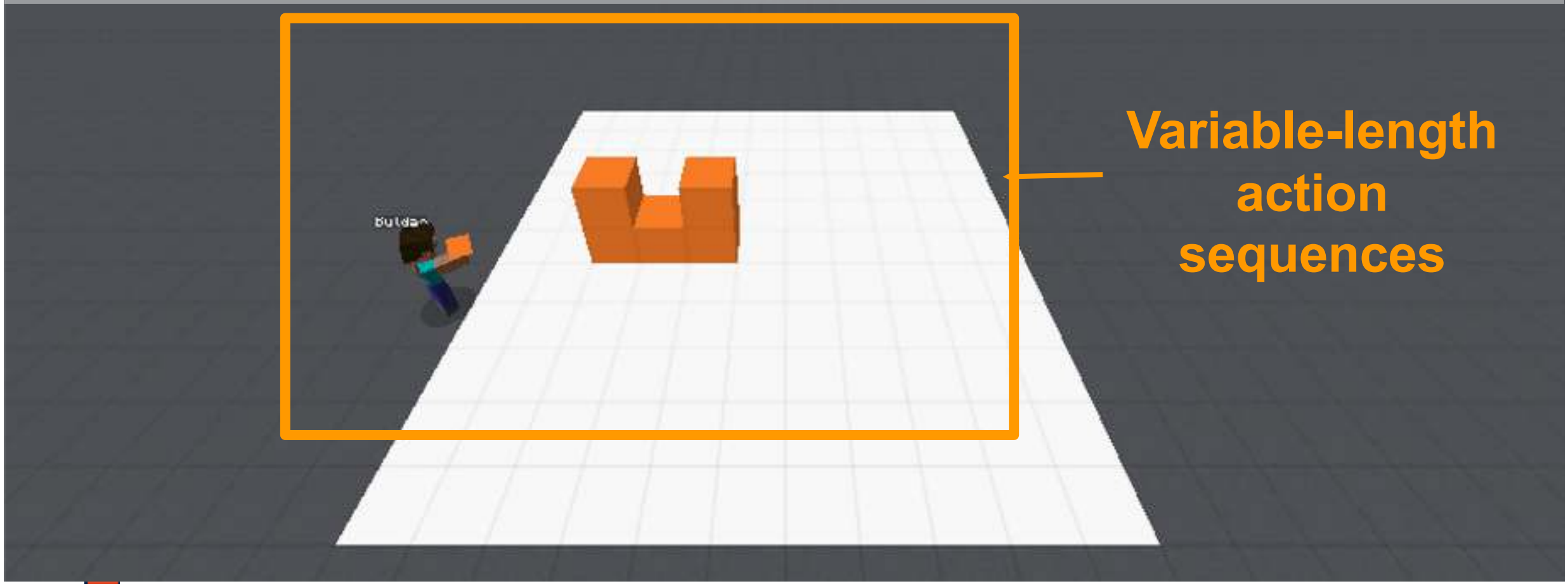
Variable-length action sequences

<Architect> go the middle and place an orange block two spaces to the left
<Architect> now make a staircase with 2 stairs left and 2 right with orange
<Architect> so it will look like a v

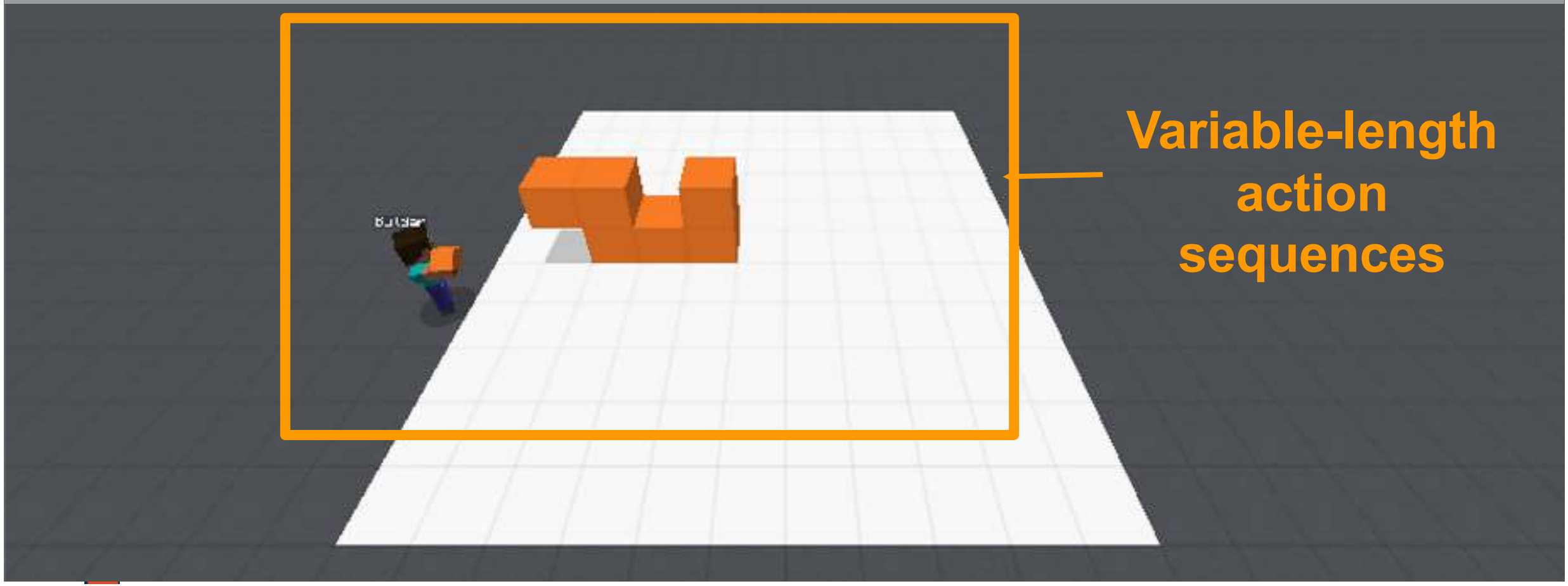


Variable-length action sequences

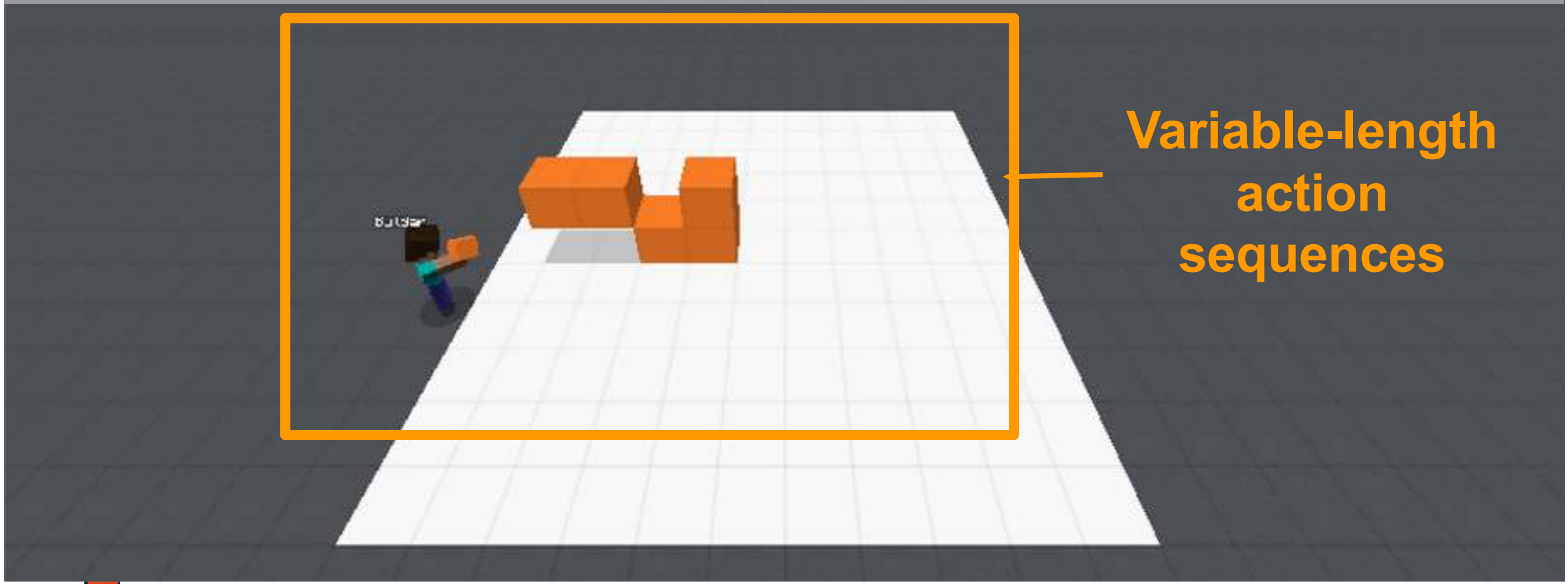
<Architect> go the middle and place an orange block two spaces to the left
<Architect> now make a staircase with 2 stairs left and 2 right with orange
<Architect> so it will look like a v



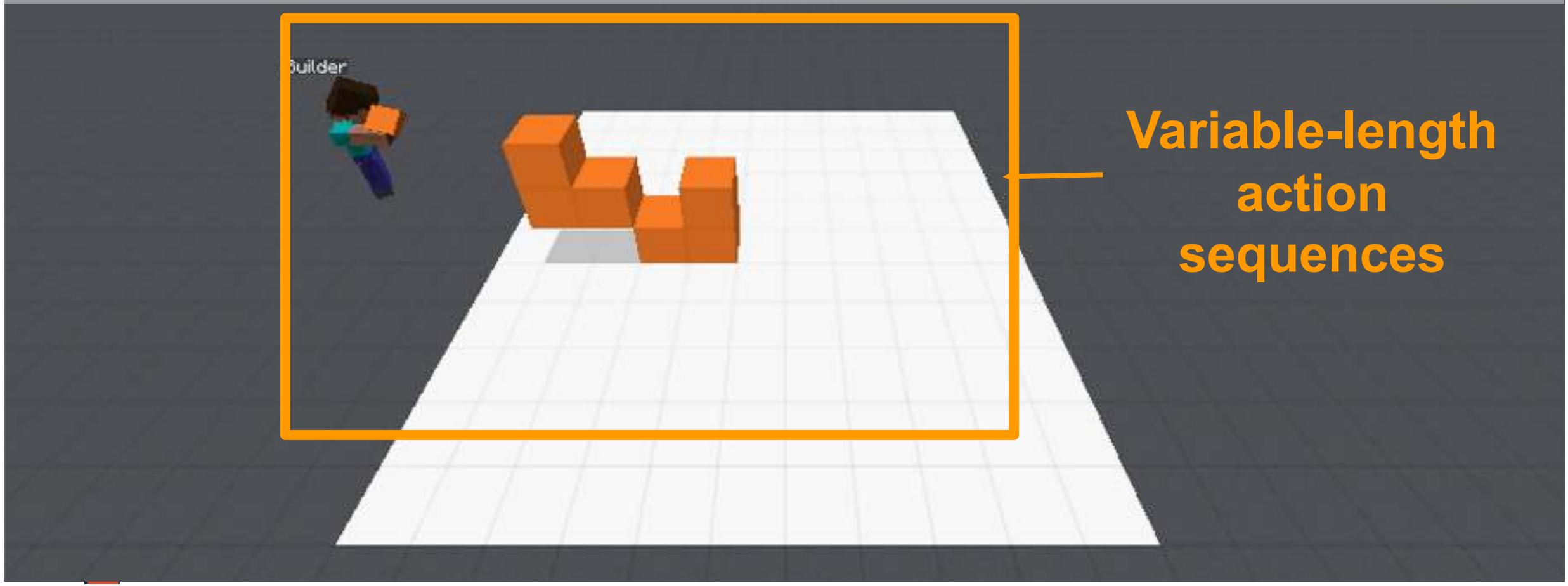
<Architect> go the middle and place an orange block two spaces to the left
<Architect> now make a staircase with 2 stairs left and 2 right with orange
<Architect> so it will look like a v



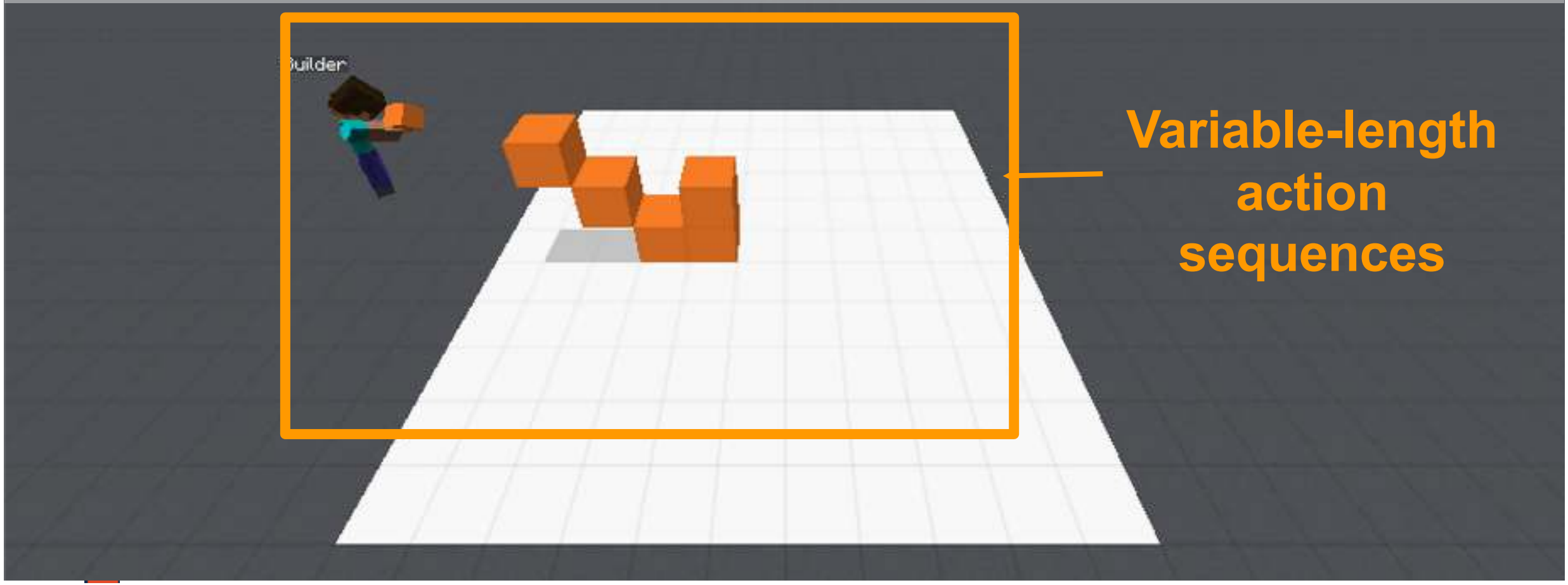
<Architect> go the middle and place an orange block two spaces to the left
<Architect> now make a staircase with 2 stairs left and 2 right with orange
<Architect> so it will look like a v



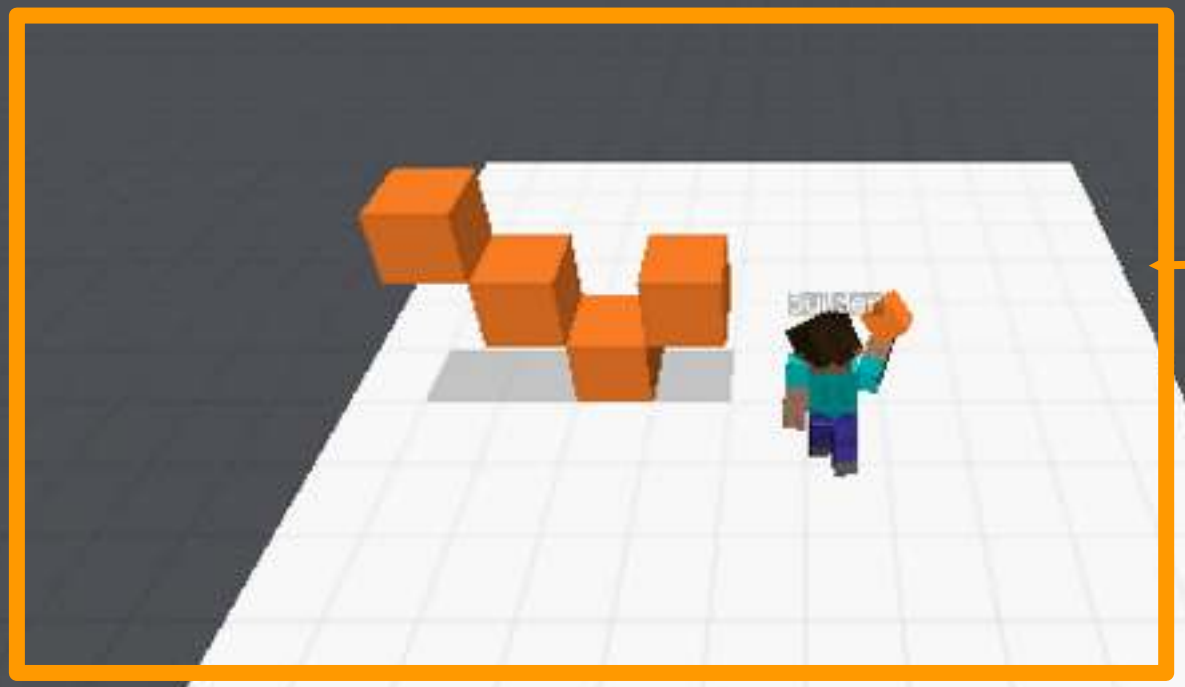
<Architect> go the middle and place an orange block two spaces to the left
<Architect> now make a staircase with 2 stairs left and 2 right with orange
<Architect> so it will look like a v



<Architect> go the middle and place an orange block two spaces to the left
<Architect> now make a staircase with 2 stairs left and 2 right with orange
<Architect> so it will look like a v

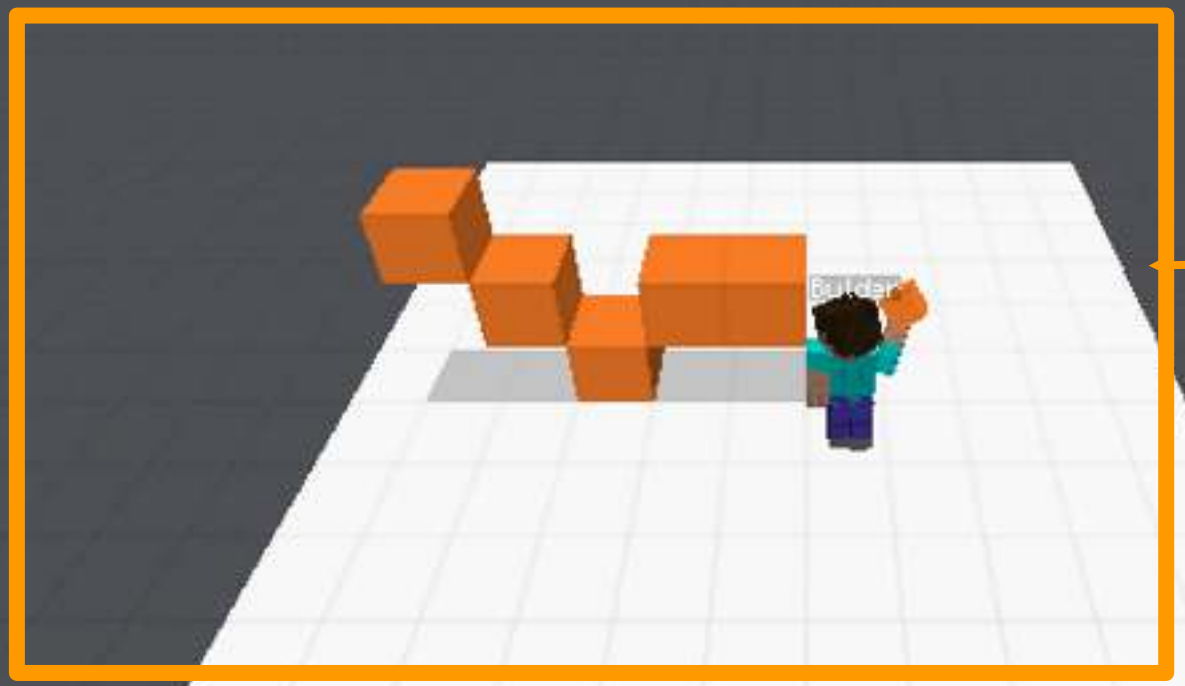


<Architect> go the middle and place an orange block two spaces to the left
<Architect> now make a staircase with 2 stairs left and 2 right with orange
<Architect> so it will look like a v



Variable-length action sequences

<Architect> go the middle and place an orange block two spaces to the left
<Architect> now make a staircase with 2 stairs left and 2 right with orange
<Architect> so it will look like a v

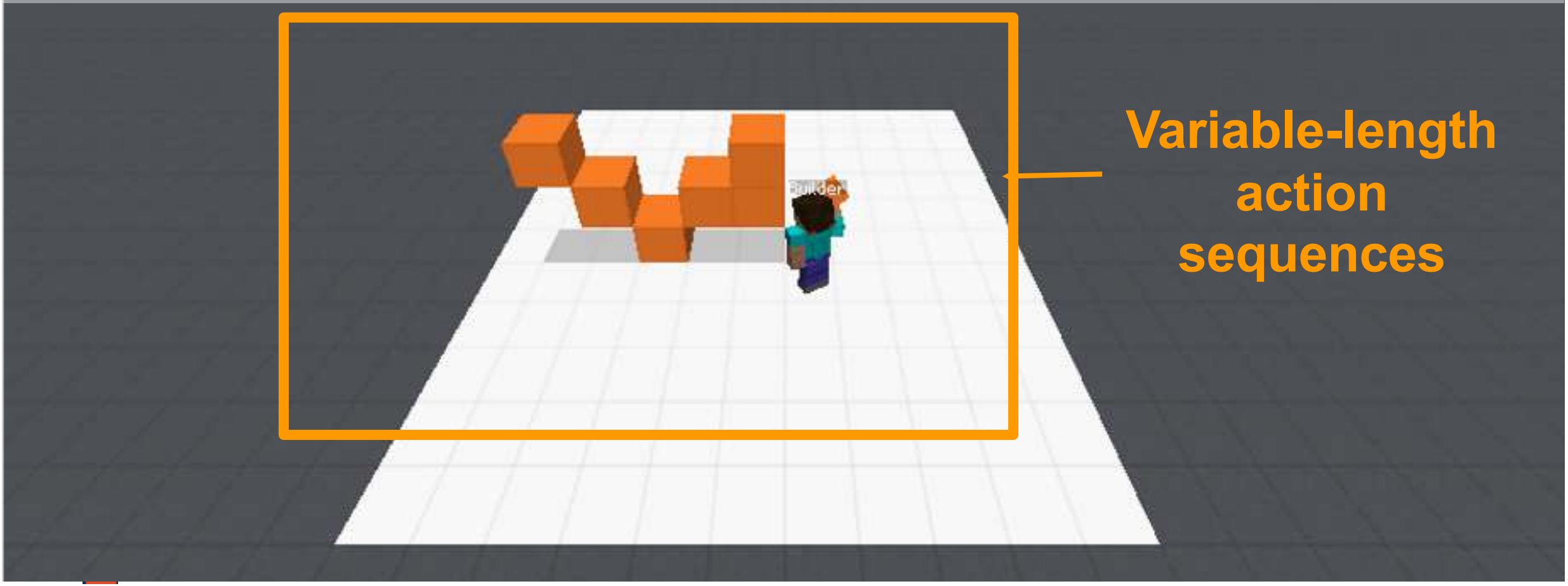


Variable-length action sequences

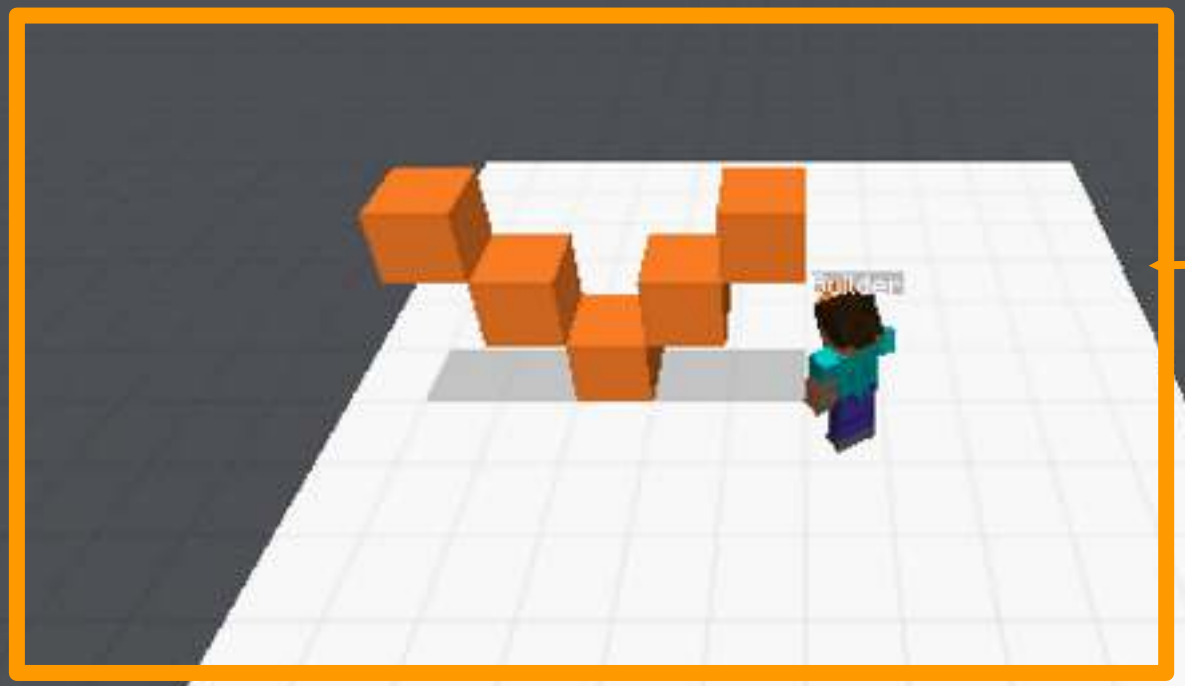
<Architect> go the middle and place an orange block two spaces to the left

<Architect> now make a staircase with 2 stairs left and 2 right with orange

<Architect> so it will look like a v

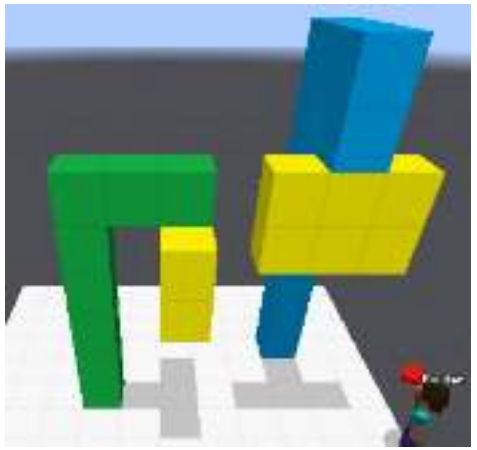
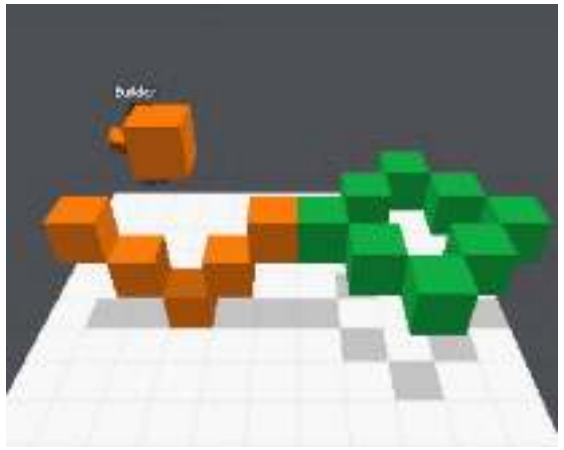
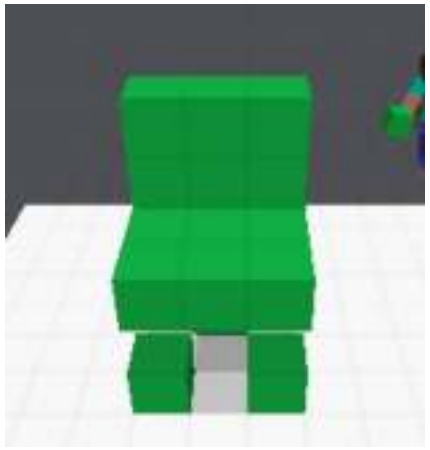
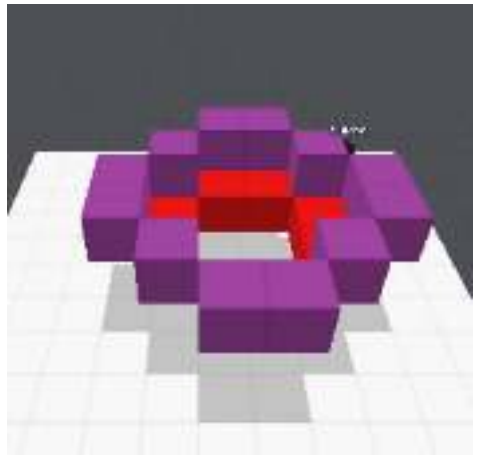
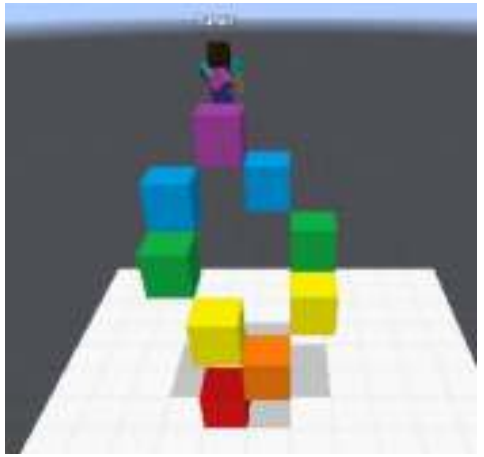


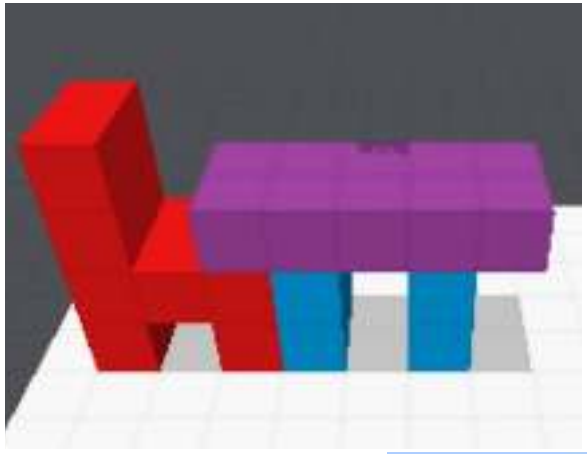
<Architect> go the middle and place an orange block two spaces to the left
<Architect> now make a staircase with 2 stairs left and 2 right with orange
<Architect> so it will look like a v

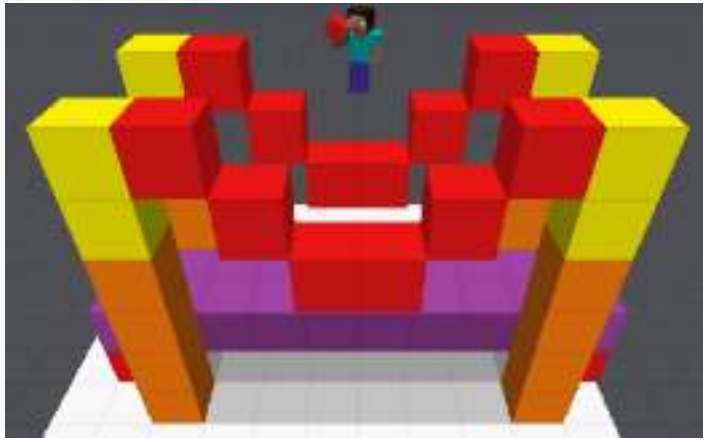
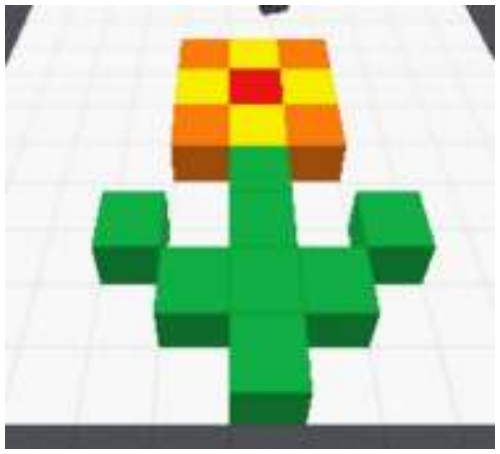
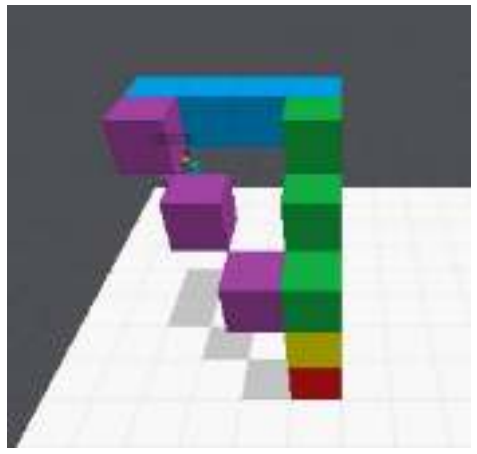
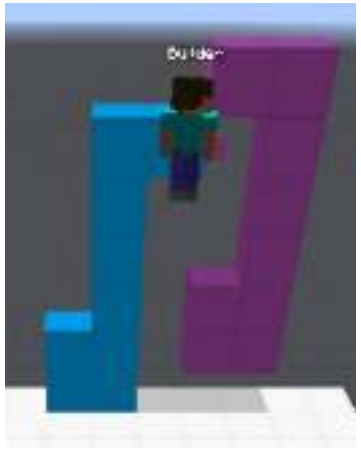


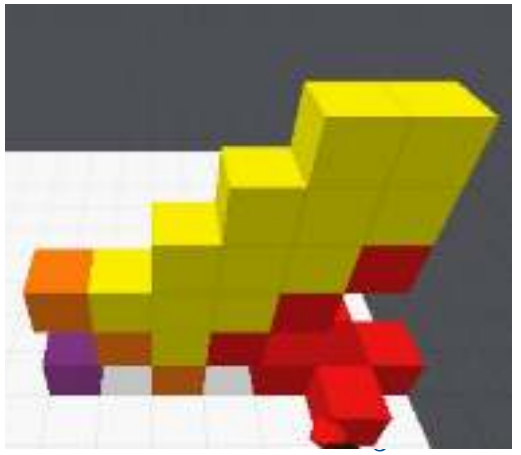
Variable-length action sequences

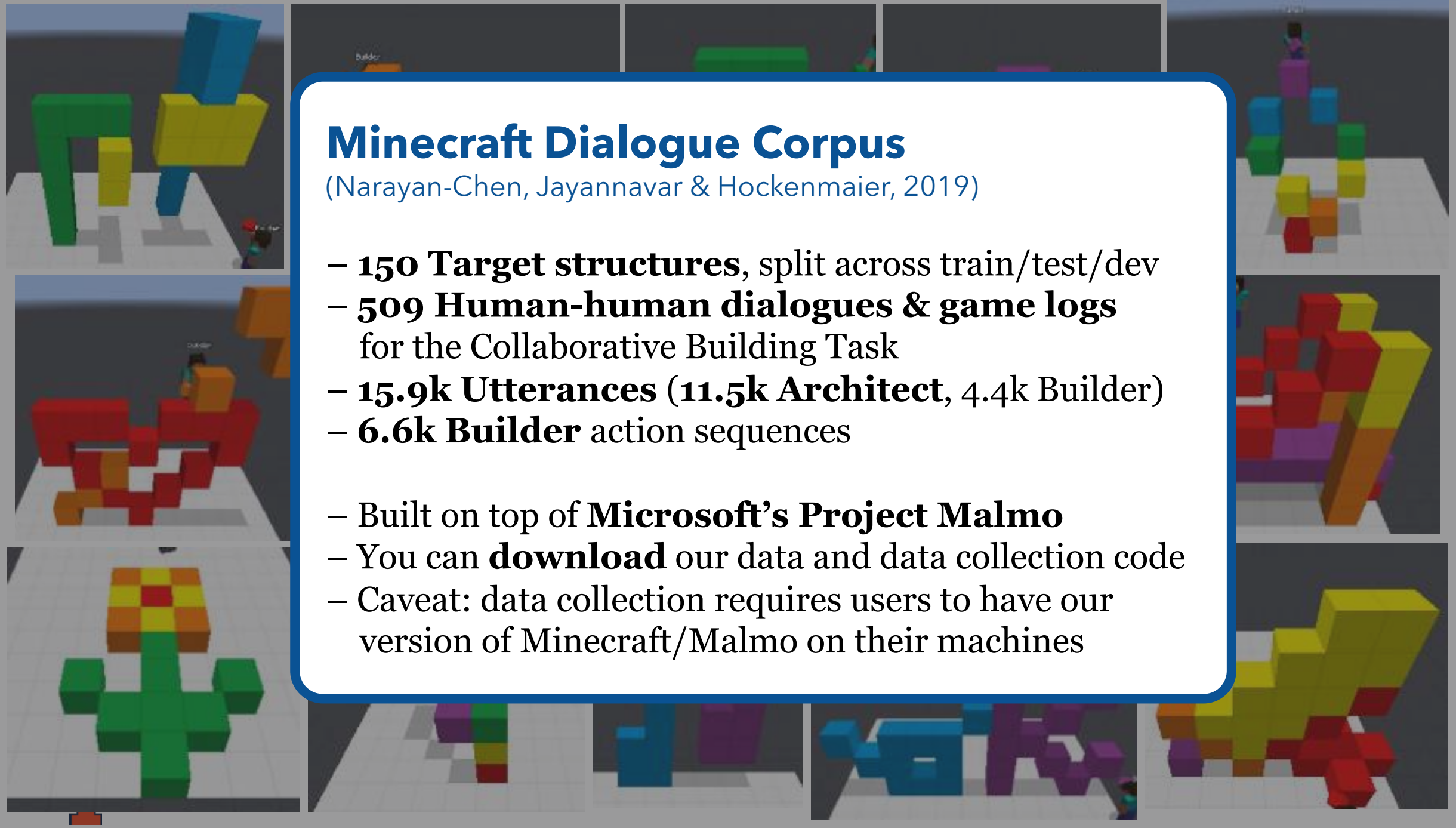
Our Dataset:
The Minecraft Dialogue Corpus







Minecraft Dialogue Corpus

(Narayan-Chen, Jayannavar & Hockenmaier, 2019)

- **150 Target structures**, split across train/test/dev
- **509 Human-human dialogues & game logs** for the Collaborative Building Task
- **15.9k Utterances** (**11.5k Architect**, 4.4k Builder)
- **6.6k Builder** action sequences

- Built on top of **Microsoft's Project Malmo**
- You can **download** our data and data collection code
- Caveat: data collection requires users to have our version of Minecraft/Malmo on their machines

How Can We Build Systems that Can Perform This Task?

How can we build systems that can perform this task?

Option 1:

Develop rich linguistic representations for this domain

Annotate the Minecraft Dialogue Corpus

Train generation and parsing models on these annotations

Develop agents that use these models

Option 2:

Train end-to-end neural models on this data

An Intermediate Linguistic Representation: AMRs for Dialogue and Spatial Relations

Bonn, Palmer, Cai, Wright-Bettner, LREC 2020

Spatial PropBank Rolesets

Rolesets do the heavy lifting in AMRs:

Multi-alias Rolesets: apply to a fixed set of synonymous spatial expressions

Roles: semantic/pragmatic roles, annotated with participants from the text

Entailments: meaning expressed by the roleset itself, no need to annotate manually

left-j
leftward-r
on_the_left-m

left-20
ARG1-SE1 entity on the left
ARG2-SE2 of what?
ARG3-ANC anchor for FoR
ARG4-AXS axis

discrete(SE1, SE2)
framework(ANC)
horizontal(AXS, ANC)

Expanded Roleset Inventory:
186 new/updated rolesets
verbs, nouns, adjectives, prepositions, adverbs, MWEs
20 new semantic/pragmatic role types

AMR Annotation

Single-sentence annotation: surface representation & frame of reference
Multi-sentence annotation: intersentential coreference & implicit arguments
Dummy AMR: maps specific spatial frameworks from dialogue onto absolute coordinate system

Now one to the left.

```
(b / be-destined-for-91
      :ARG1 (t / thing :quant 1)
      :ARG2 (s2 / space
            :ARG1-of (l / left-20
                  :ARG2 [implicit: previous block]
                  :ARG3 (c /cartesian-framework-91
                        :ARG1 (b2 / builder))))
      :time (n / now))
```

Inferred concept

Dialogue-level coreference

Frame of Reference co-refers to dummy AMR

Translation from surface 'left' → absolute form:
yaw= 88.2 → B is facing EAST → "left" = absolute NORTH

Annotation Statistics:

243 full dialogues
7255 dialogue sentences (+ 11,000 auto-generated non-dialogue construction AMRs)

How can we build agents that can perform this task?

Option 1:

Develop rich linguistic representations for this domain

Annotate the Minecraft Dialogue Corpus

Train generation and parsing models on these annotations

Develop agents that use these models

Option 2:

Train end-to-end neural models on this data

Starting Point for Architect: Utterance Generation

(Narayan-Chen, Jayannavar, Hockenmaier, ACL 2019)

Architect: Tasks and Challenges

Give clear and correct instructions in a changing environment

A. needs to identify **next steps** for B.
A. needs to **align target and build** region
A. needs to adapt to **B's current position**
A. needs to **identify mistakes** made by B.

Answer Builder's questions

Interrupt the Builder to correct mistakes

A. should **respond in real time** (no turns)

Architect Utterance Generation Task

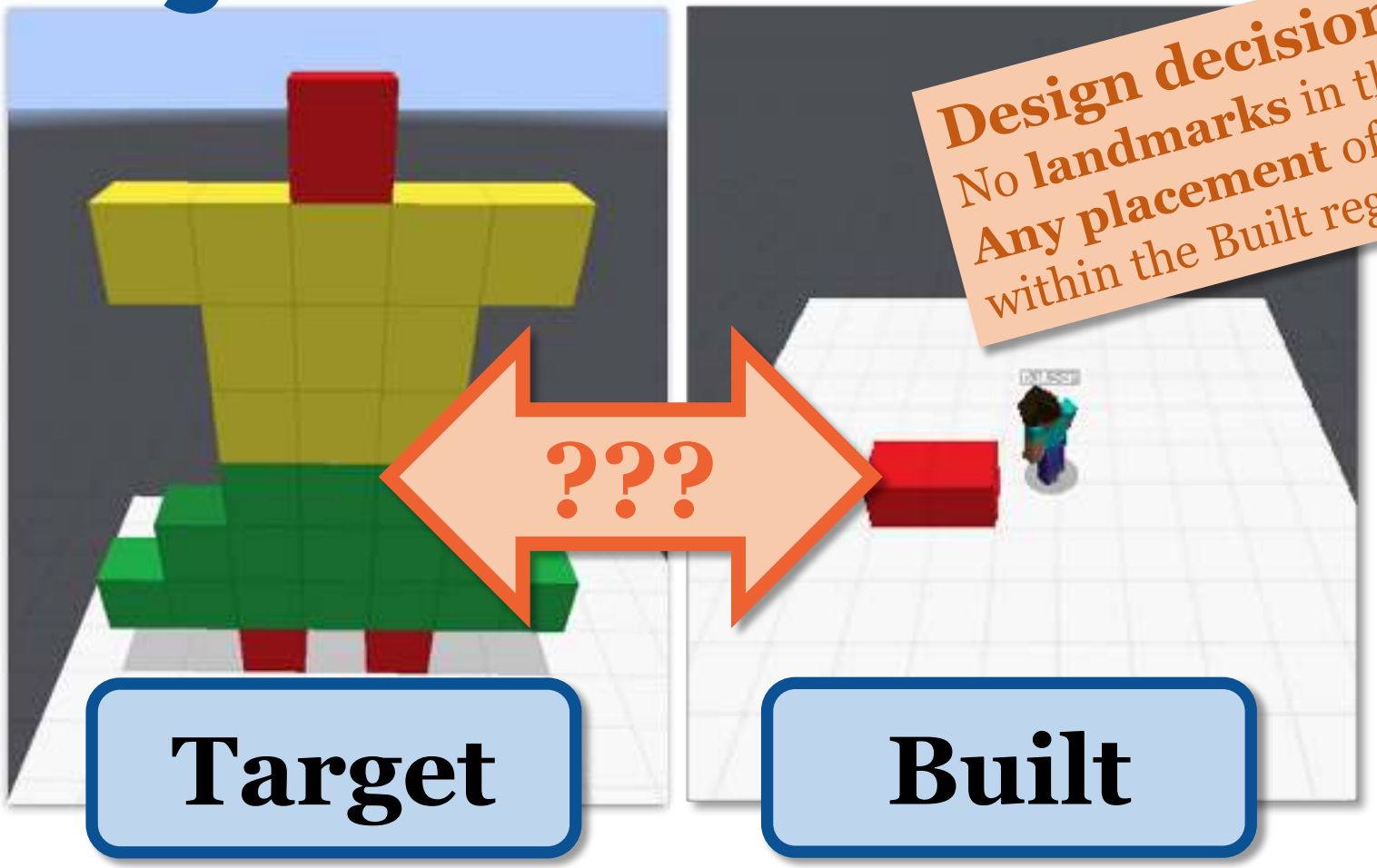
Generate a suitable Architect utterance
for a game state in a human-human game
when the human Architect said something.

Ignores **real-time** aspect
(when to speak)

Ignores **overall task** completion
(how to maintain a whole conversation)

Allows us to use supervised learning to develop **baseline models**

Modeling the World State: Align Target and Built structures



Modeling the World State naively with Block Counters

Global Block counters (one 18-dimensional vector)

For each of the 6 colors: #blocks to be **added**, **added next**, and **removed**

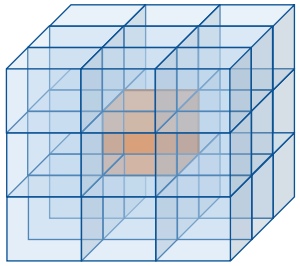
Averaged over all optimal alignments of built to target.

Add	Add Next	Remove	Add	Add Next	Remove	Add	Add Next	Remove	Add	Add Next	Remove	Add	Add Next	Remove	Add	Add Next	Remove

Local Block Counters (concatenate 27 block counters)

Separate counters for each cell in the **3×3×3 cube around the last cell** the Builder touched.

To capture the Builders' current perspective, the order of cells depends on the Builder's current position, pitch and yaw.

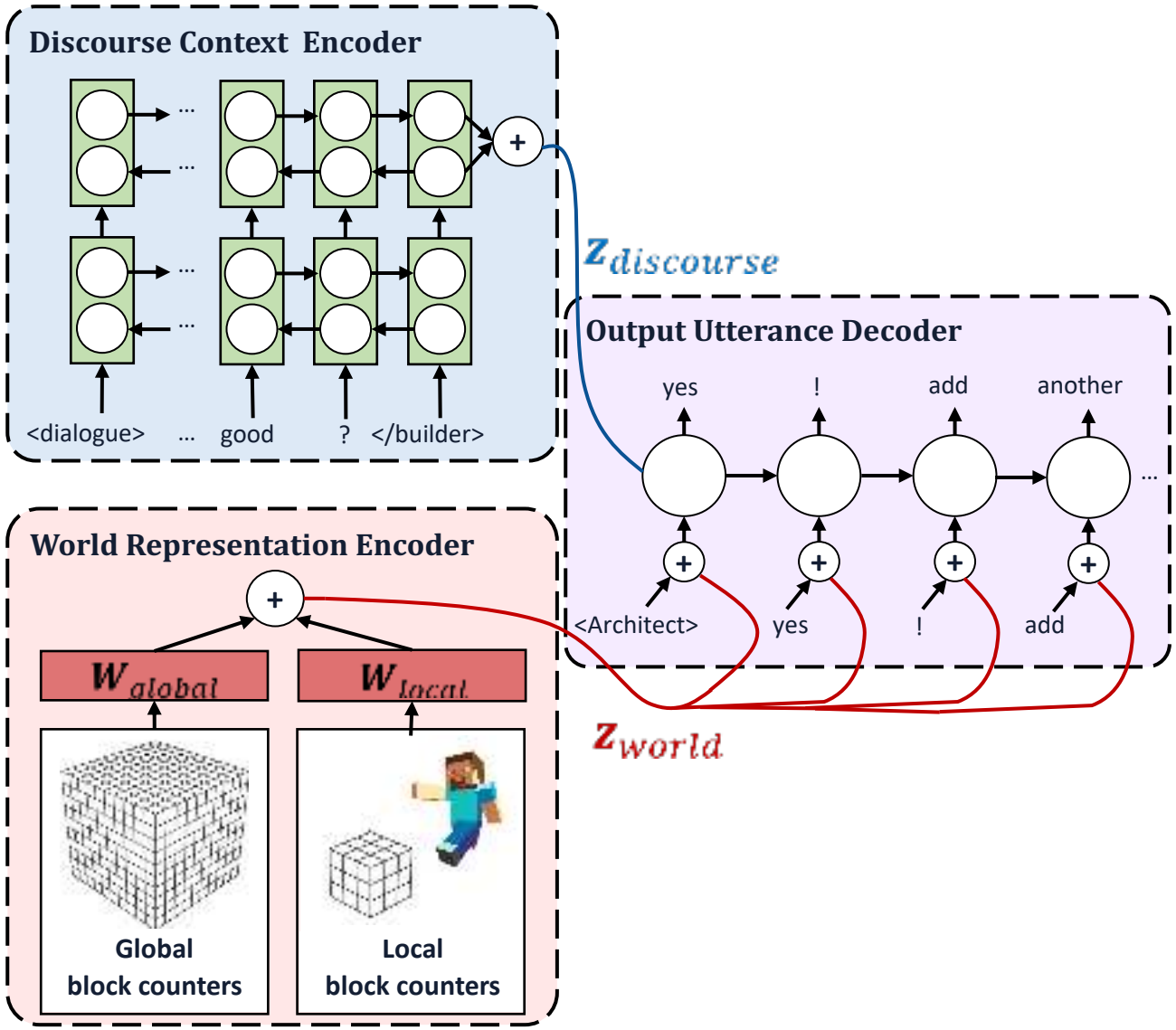


Our Model

Discourse Context encoder:
biGRU over previous dialogue with Glove embeddings

World Context Encoder:
$\mathbf{W}_{global}$: Global Block counters
$\mathbf{W}_{local}$: Local Block counters

Output utterance decoder:
Reads block counter embeddings (and last token) at each time step



Automatic Evaluation

Automatic Evaluation	BLEU-1
seq2seq	15.3
Block Counter	**15.7**

Block Counter model gives a **minor improvement in BLEU-1.**

Automatic Evaluation

Automatic Evaluation	BLEU-1	Spatial P/R
seq2seq	15.3	**9.3**/8.6
Block Counter	**15.7**	8.7/8.7

Block Counter model gives a **minor improvement in BLEU-1.**
Block Counter model has **slightly lower performance on spatial terms.**

Automatic Evaluation

Automatic Evaluation	BLEU-1	Spatial P/R	Color P/R
seq2seq	15.3	**9.3**/8.6	8.1/17.0
Block Counter	**15.7**	8.7/8.7	**14.9/28.7**

Block Counter model gives a **minor improvement in BLEU-1.**
Block Counter model has **slightly lower performance on spatial terms.**
Block Counter model has **much better precision and recall of color terms.**

Human Evaluation

How **correct** are the generated utterances
(wrt. **current game state and target**)?
Correct utterances are more likely to lead to **task completion**.

	Fully correct	Partially correct	Incorrect
Human (ceiling)	**89.0%**	**0.0%**	**0.0%**

Most human utterances are **fully correct**
(remainder: correctness can't be assessed, e.g. in chit-chat)

Human Evaluation

How **correct** are the generated utterances
(wrt. **current game state and target**)?
Correct utterances are more likely to lead to **task completion**.

	Fully correct	Partially correct	Incorrect
Human (ceiling)	**89.0%**	**0.0%**	**0.0%**
seq2seq (baseline)	14.0%	28.0%	**48.0%**

Almost half of the **baseline model**'s utterances are incorrect.

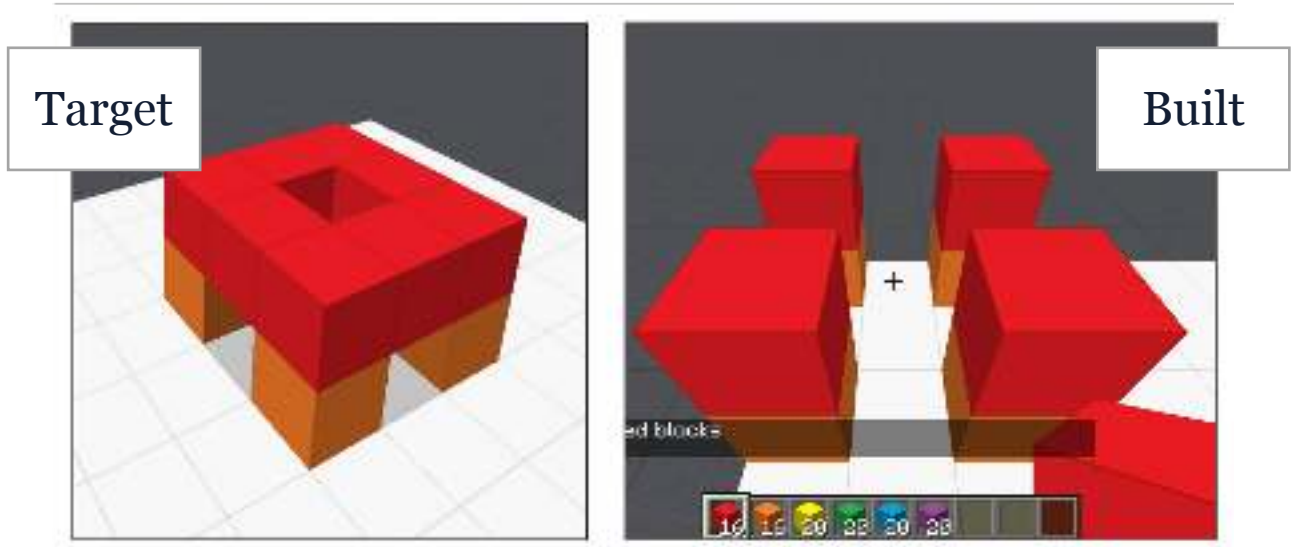
Human Evaluation

How **correct** are the generated utterances
(wrt. **current game state and target**)?
Correct utterances are more likely to lead to **task completion**.

	Fully correct	Partially correct	Incorrect
Human (ceiling)	**89.0%**	**0.0%**	**0.0%**
seq2seq (baseline)	14.0%	28.0%	48.0%
Block Counters	**25.0%**	**36.0%**	**32.0%**

The **Block Counter** Model produces **significantly more fully/partially correct utterances** and **significantly fewer incorrect ones** than the baseline (even if it is still pretty far from human performance)

What can the neural Architect do?

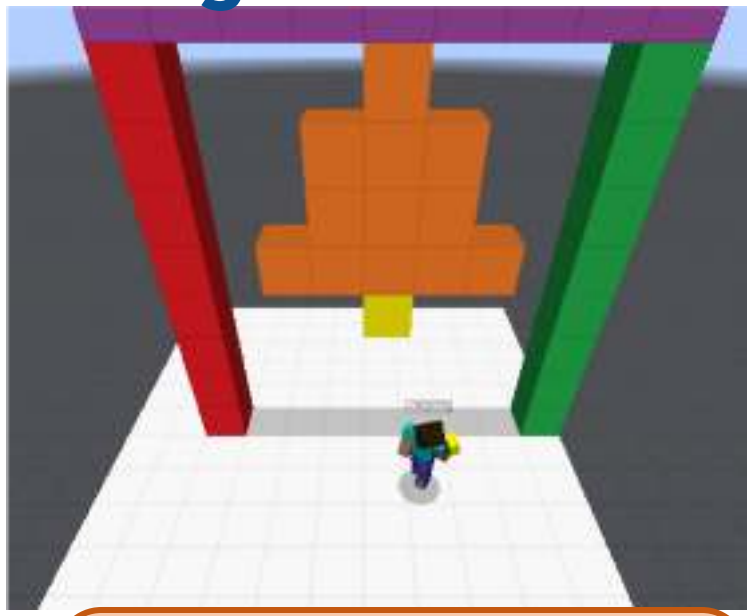


Builder has just placed the red block in the top right corner

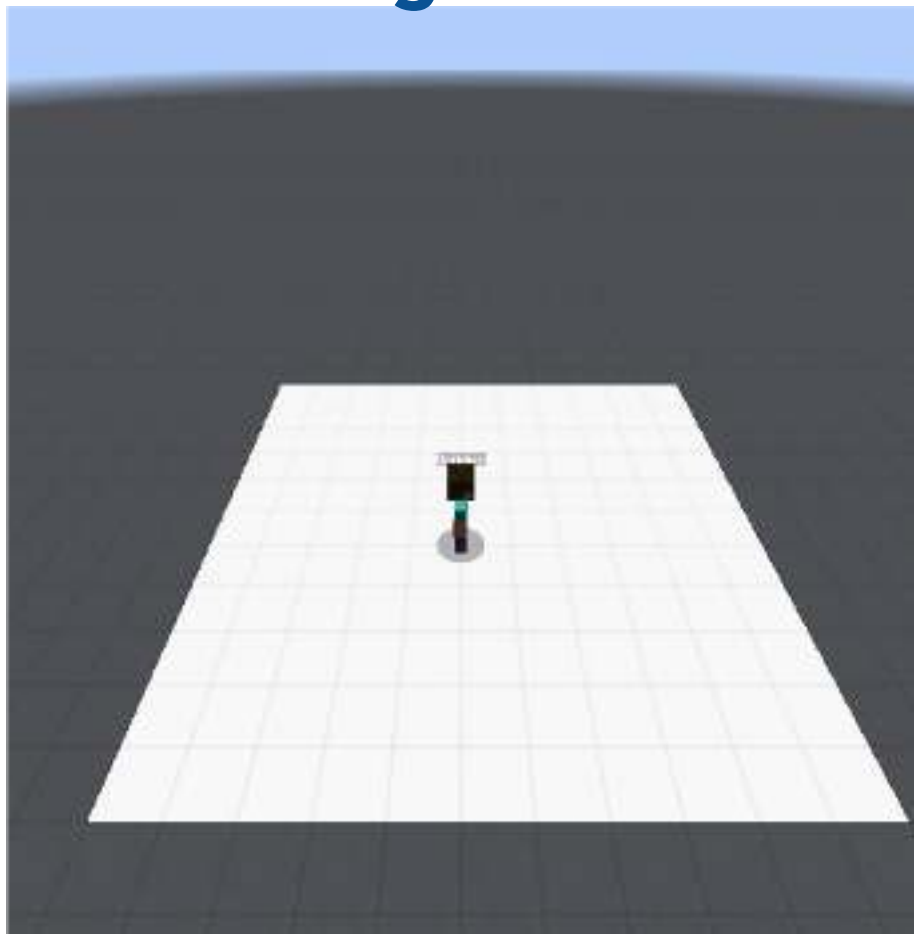
A: *"perfect! now place a red block to the left of that"*

The neural architect gives natural, **fluent block-by-block instructions** that contain **color terms** and **spatial relations**

Target structure



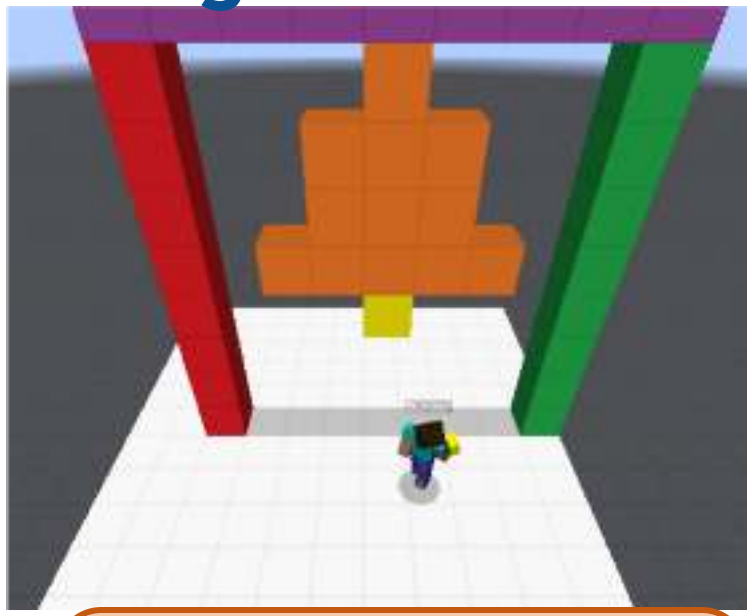
Current game state



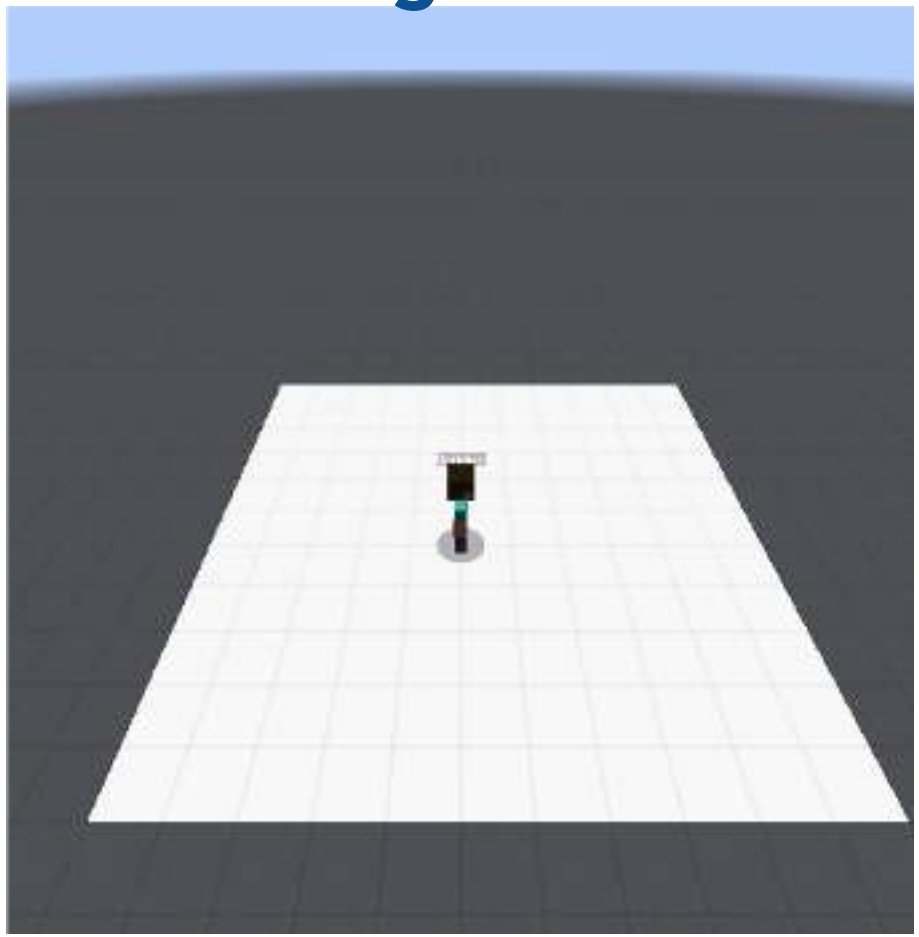
Blue: Model Architect

Model A: okay , we 'll start with a row of three red blocks , place a red block in front of you

Target structure



Current game state

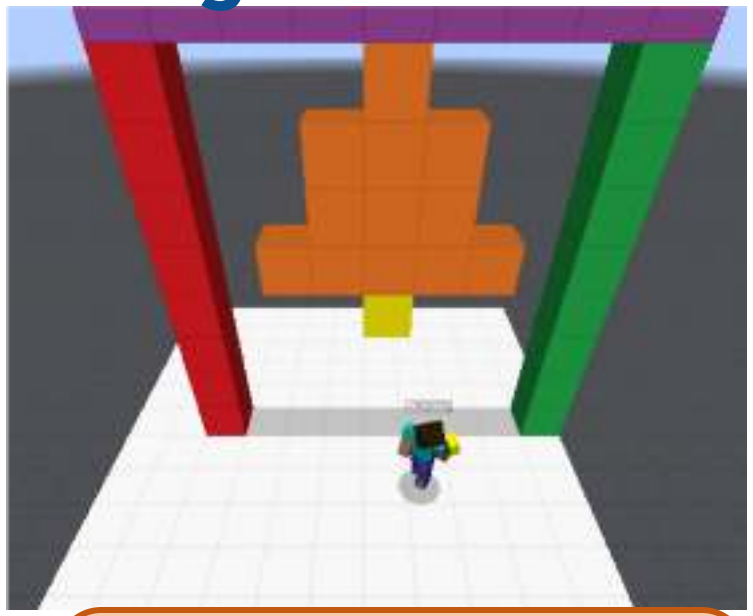


Blue: Model Architect
Red: Mistakes

Model A: okay , we 'll start with a row of three red blocks , place a red block in front of you

The block counters model has no access to **complex shapes** (*rows* vs. *towers/columns*).

Target structure



Current game state

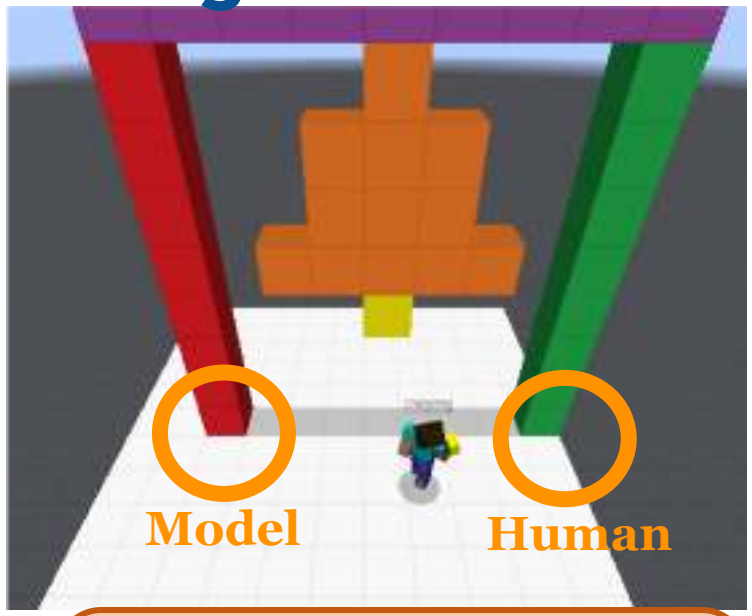
Humans give **much more complex instructions.** (“*belltower*”)

Blue: Model Architect
Red: Mistakes
Green: Human Architect

Model A: okay , we 'll start with a row of three red blocks , place a red block in front of you

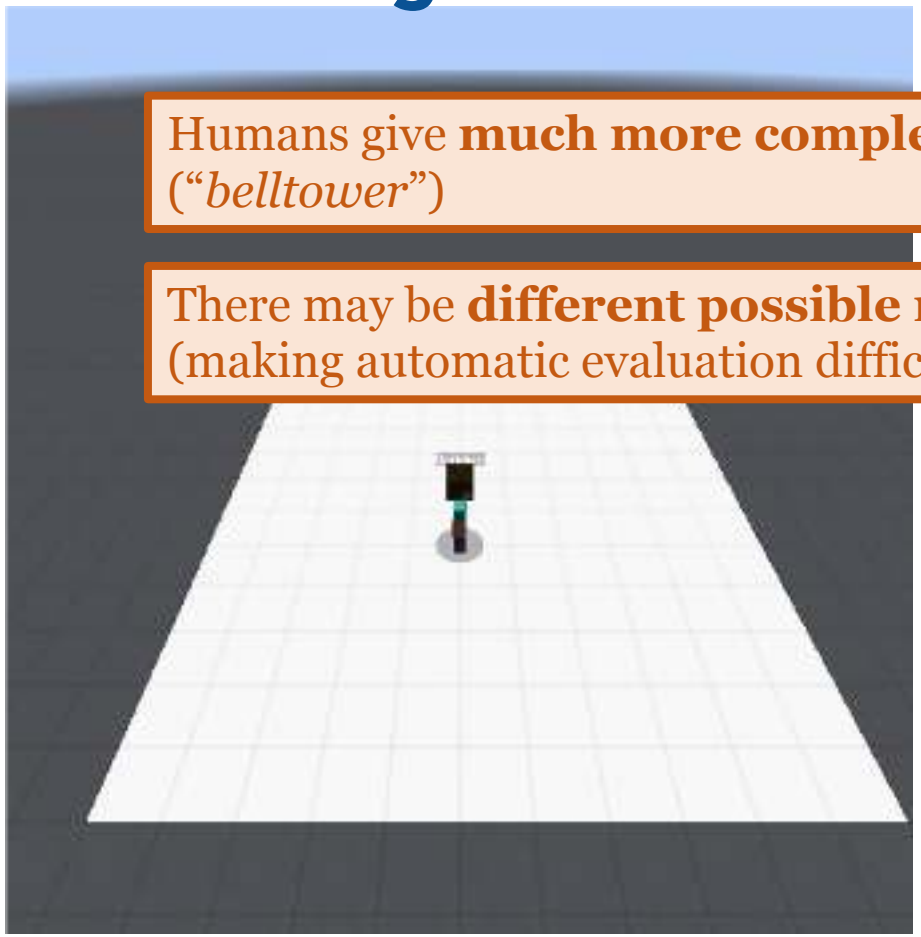
Human A: hello builder , i will tell you this. it appears we are creating a belltower . but first i will start with step by step instructions. we will start with green blocks

Target structure



Blue: Model Architect
Red: Mistakes
Green: Human Architect

Current game state



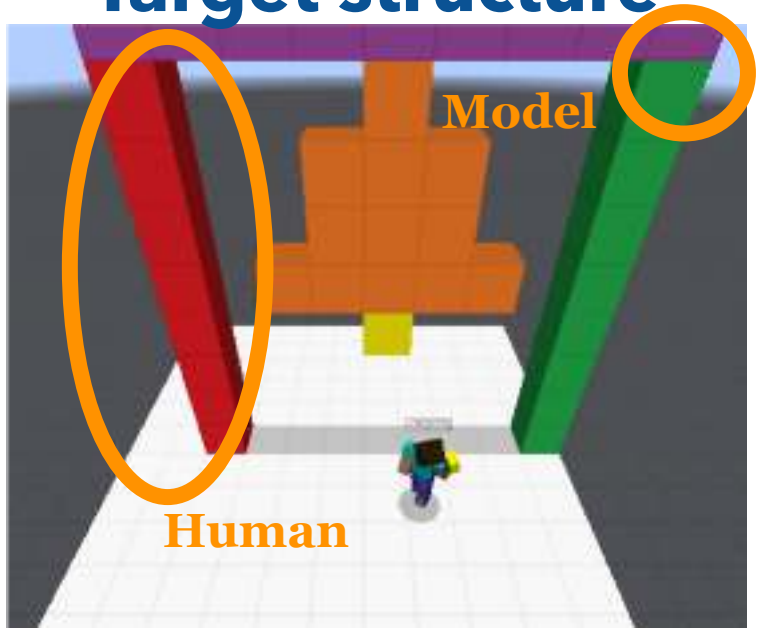
Humans give **much more complex instructions.** (“*belltower*”)

There may be **different possible next actions** (making automatic evaluation difficult)

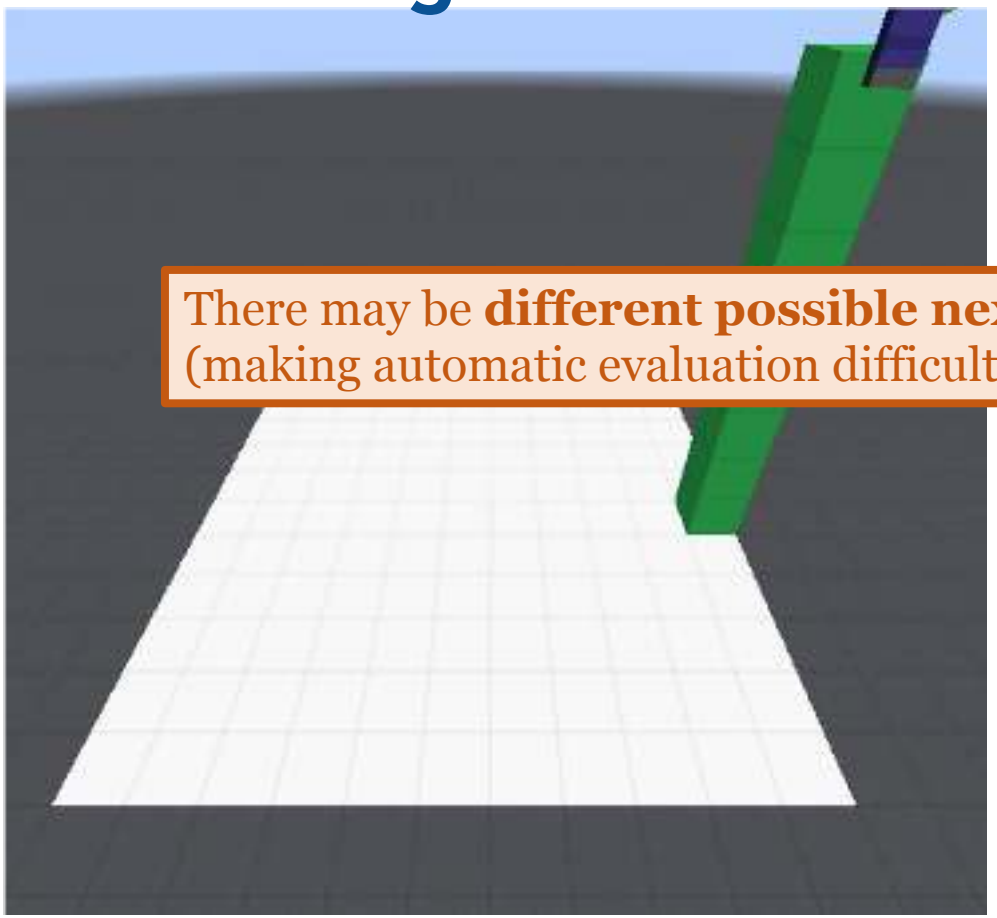
Model A: okay , we 'll start with a row of three red blocks , place a red block in front of you

Human A: hello builder , i will tell you this. it appears we are creating a belltower . but first i will start with step by step instructions. we will start with **green blocks**

Target structure



Current game state



There may be **different possible next actions** (making automatic evaluation difficult)

Human B: is this good?

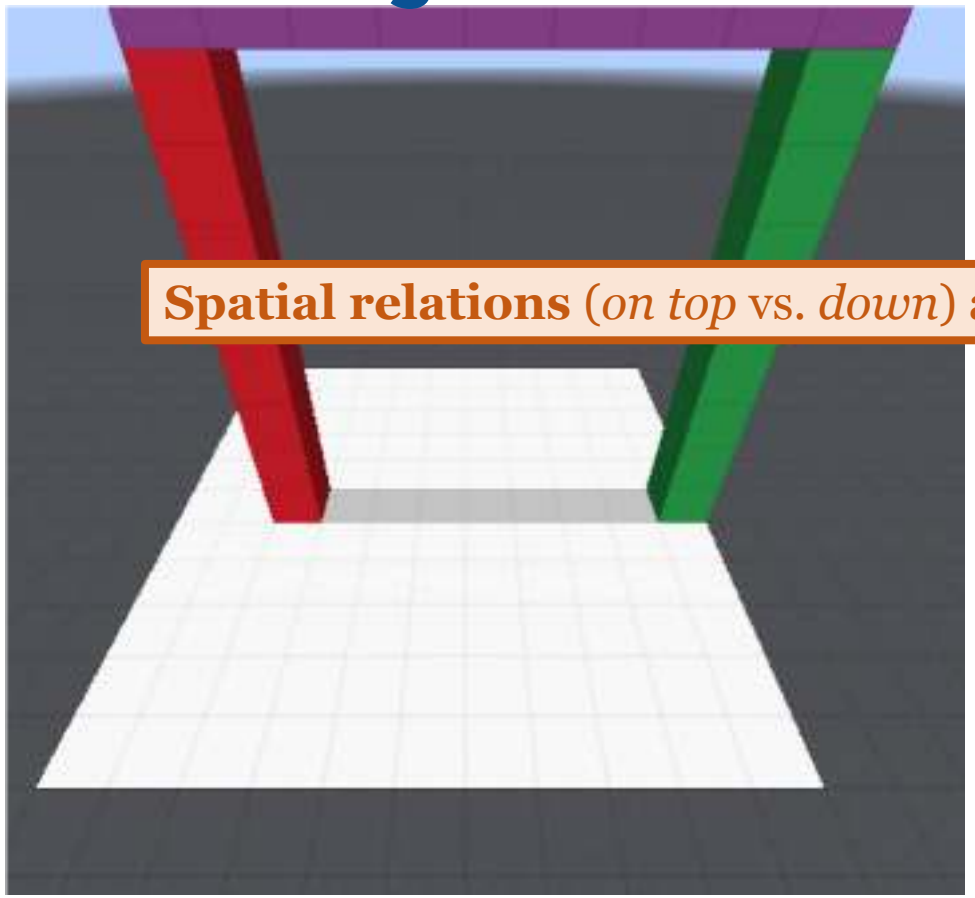
Human A: yes , one moment

Model A: perfect ! now put a purple block on top of that

Human A: on the 8th block to the left of the green block , please create **the same structure with red blocks**

Target structure

Current game state



Spatial relations (*on top* vs. *down*) are difficult

Human A: excellent builder

Model A: now place an orange block on top of that

Human A: now we must create the bell . please start by extending 4 orange blocks **down** from the middle purple block , as if it were hanging

Starting Point for Builder: Action Prediction

(Jayannavar, Narayan-Chen, Hockenmaier, ACL 2020)

Builder: Task and Challenges

Understand and execute instructions

B. needs to understand **descriptions of structures**
B. needs to understand **spatial relations**
B. needs to understand utterances in the **current context**

*Execution: **place and remove blocks** in the 11×9×11 build region*

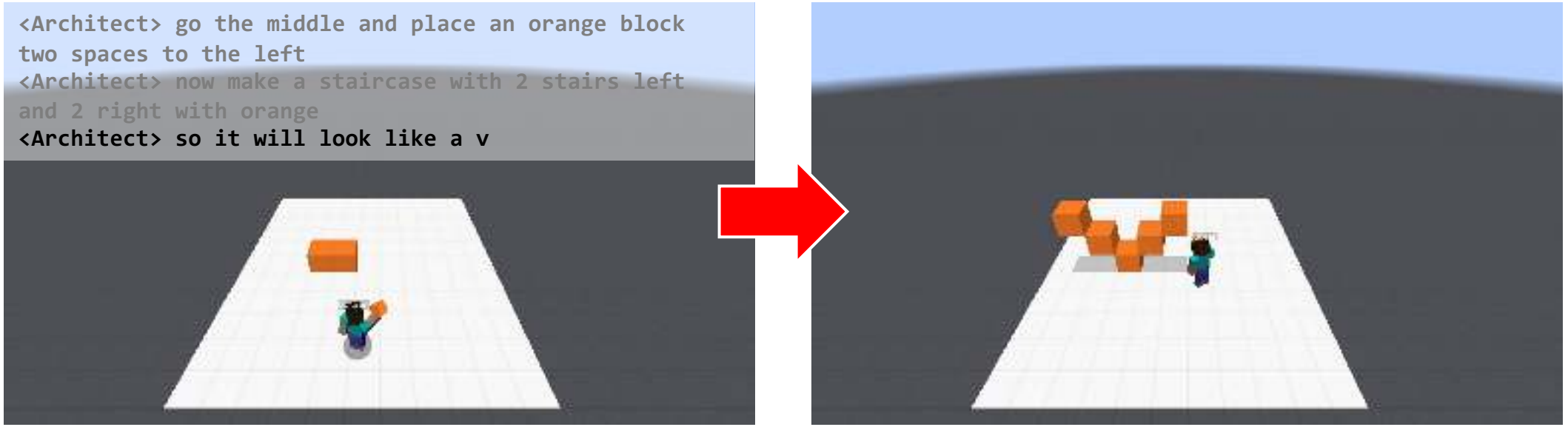
Ask clarification questions as needed

B. needs to know what information is **missing or unclear**
B. needs to know when instructions **can't be executed**

Future work:
Requires execution model

The Builder Action Prediction (BAP) Task

Predict the **sequence of actions** (block placements and/or removals) that a Builder performed at a particular point in a human-human game



BAP Task Challenges

Situated Dialogue

Understanding instructions requires **dialogue and world context**

A's instructions are spread across turns

B's perspective changes with movement

B's actions change the environment

Action Prediction

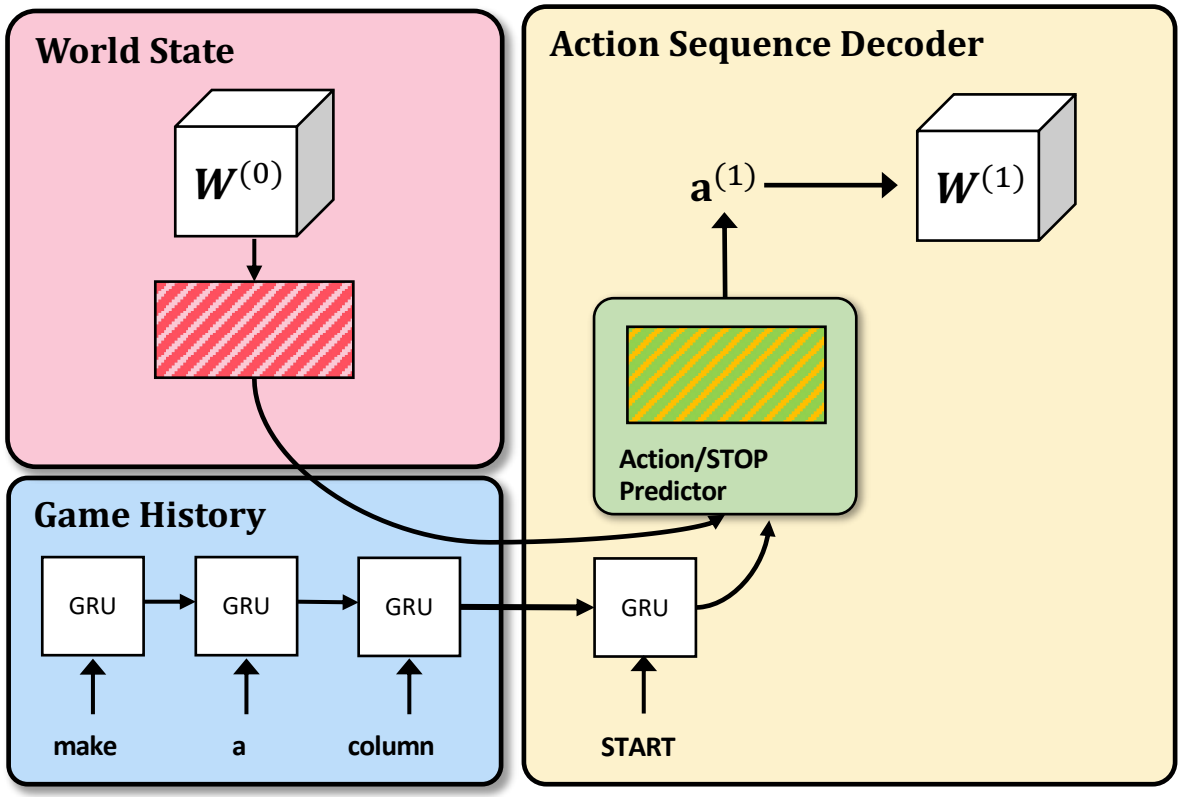
Large action space of 7,623 actions

Floating blocks require nonmonotonic action sequences where placement is followed by removal

Training sequences are noisy because humans are prone to mistakes and misunderstanding (and architects interrupt)

Our Model

Encoder-decoder network with GRU backbone



Inputs:

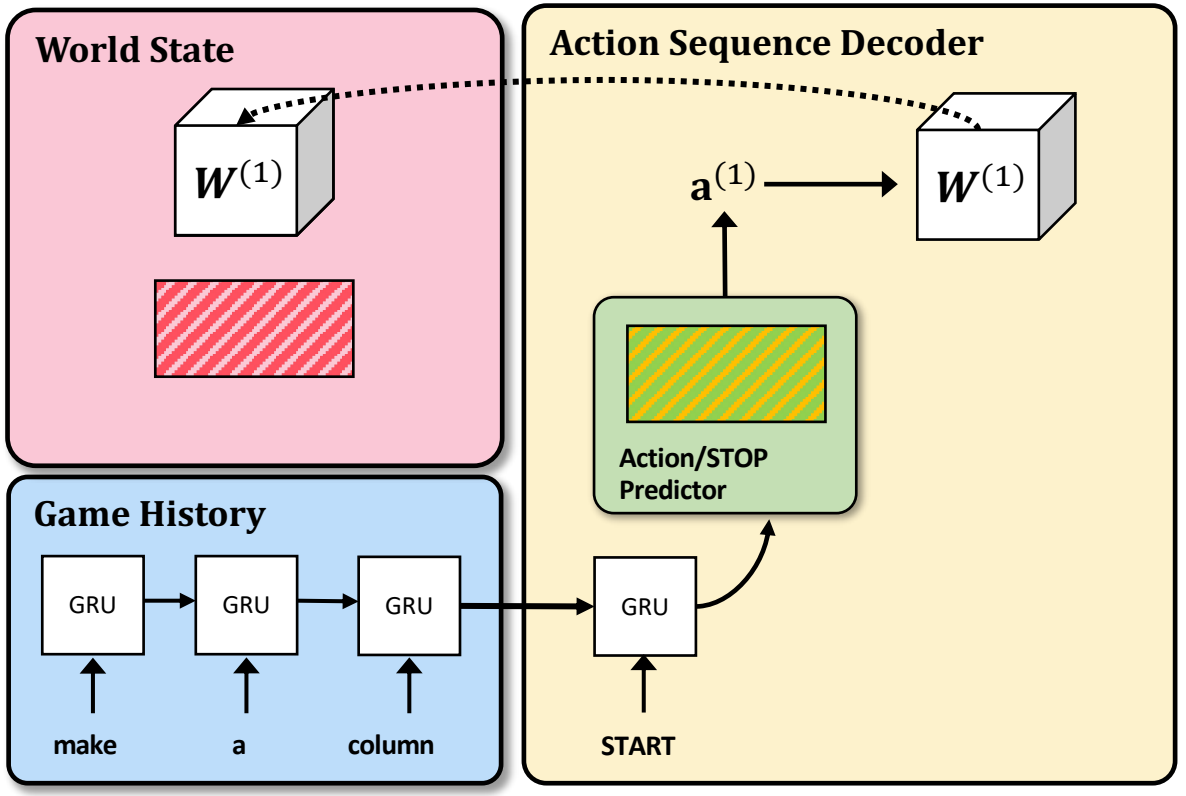
- Game history up to $t = 0$
- World state grid $W^{(0)}$

Predicts:

- Sequence of $\mathbf{B}$ actions $\mathbf{a}^{(0)} \ldots \mathbf{a}^{(t+1)}$ with $\mathbf{a}^{(0)} = \texttt{START}$

Our Model

Encoder-decoder network with GRU backbone



Inputs:

Game history up to $t = 0$

World state grid $W^{(0)}$

Predicts:

Sequence of $\mathbf{B}$ actions $\mathbf{a}^{(0)} \ldots \mathbf{a}^{(t+1)}$ with $\mathbf{a}^{(0)} = \texttt{START}$

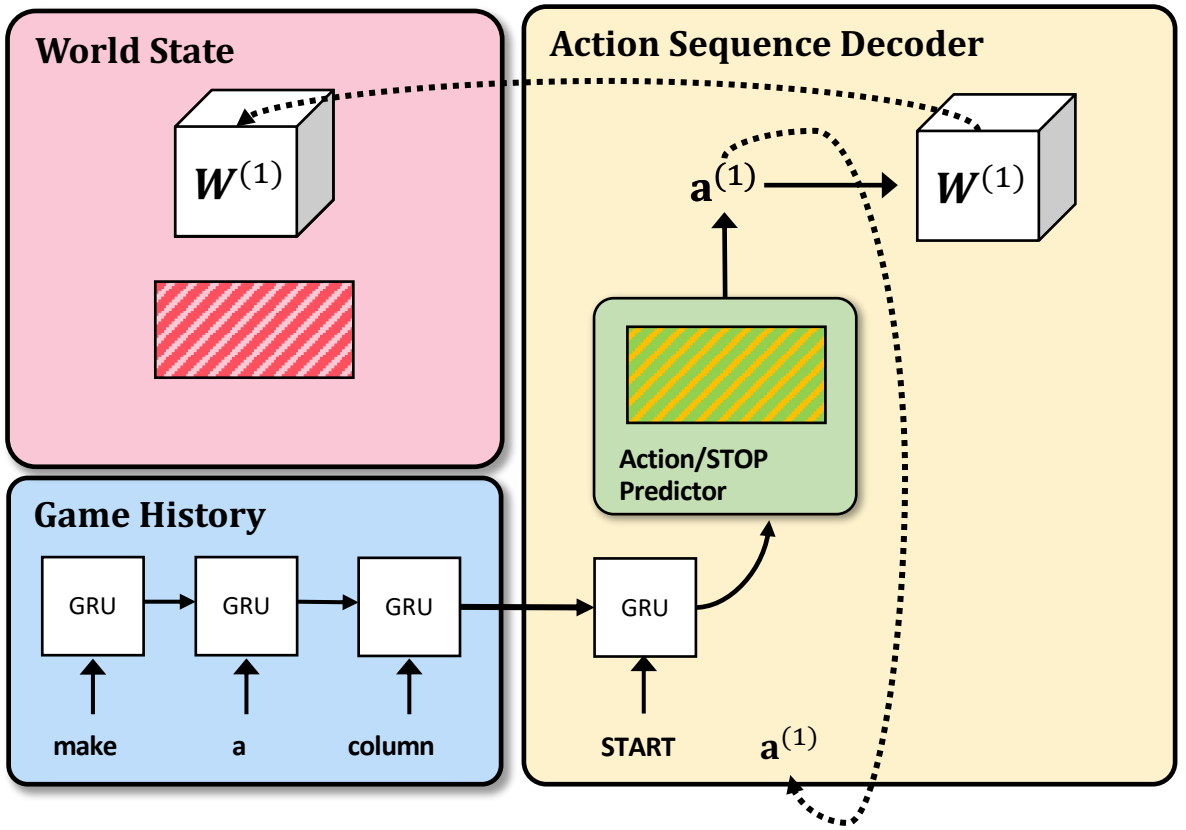
Our Model

Encoder-decoder network with GRU backbone

Inputs:

Game history up to $t = 0$

World state grid $W^{(0)}$



Predicts:

Sequence of **B** actions $\mathbf{a}^{(0)} \ldots \mathbf{a}^{(t+1)}$ with $\mathbf{a}^{(0)}$ = START

Our Model

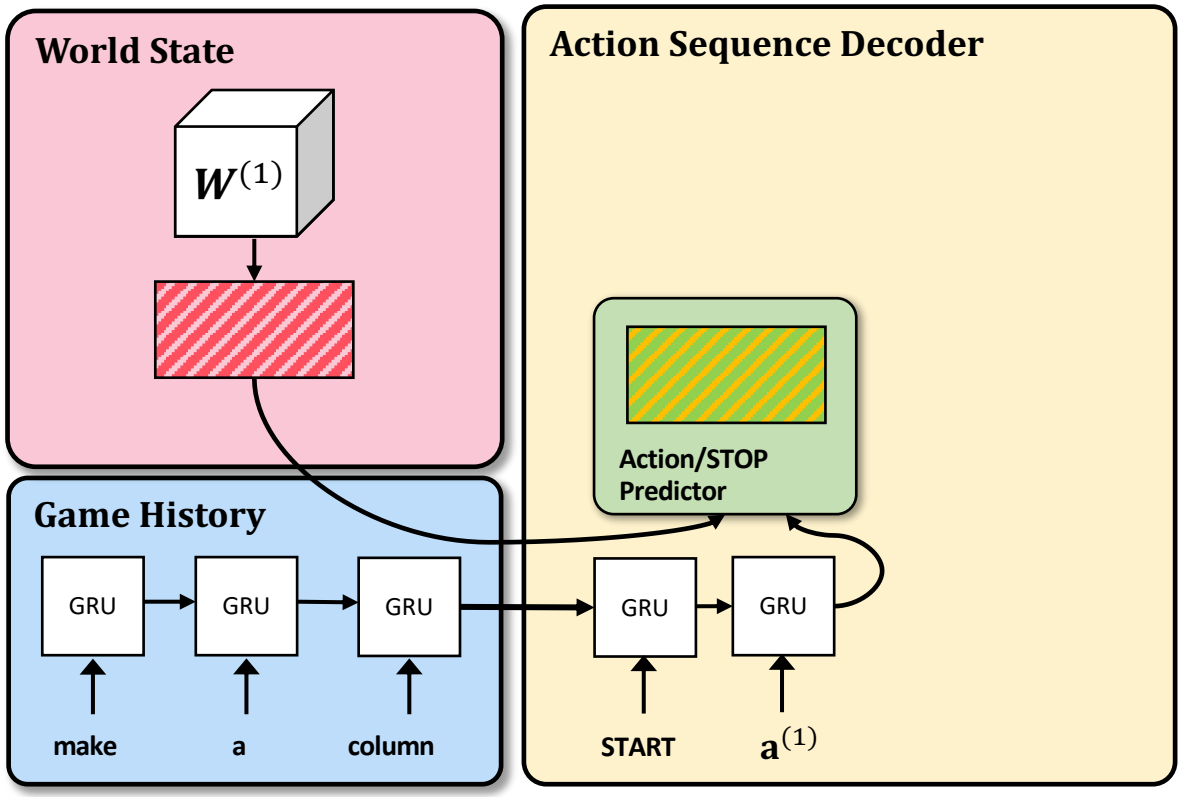
Encoder-decoder network with GRU backbone

Inputs:

- Game history up to $t = 0$
- World state grid $W^{(0)}$

Predicts:

- Sequence of **B** actions $\mathbf{a}^{(0)} \ldots \mathbf{a}^{(t+1)}$ with $\mathbf{a}^{(0)} = \texttt{START}$



Our Model

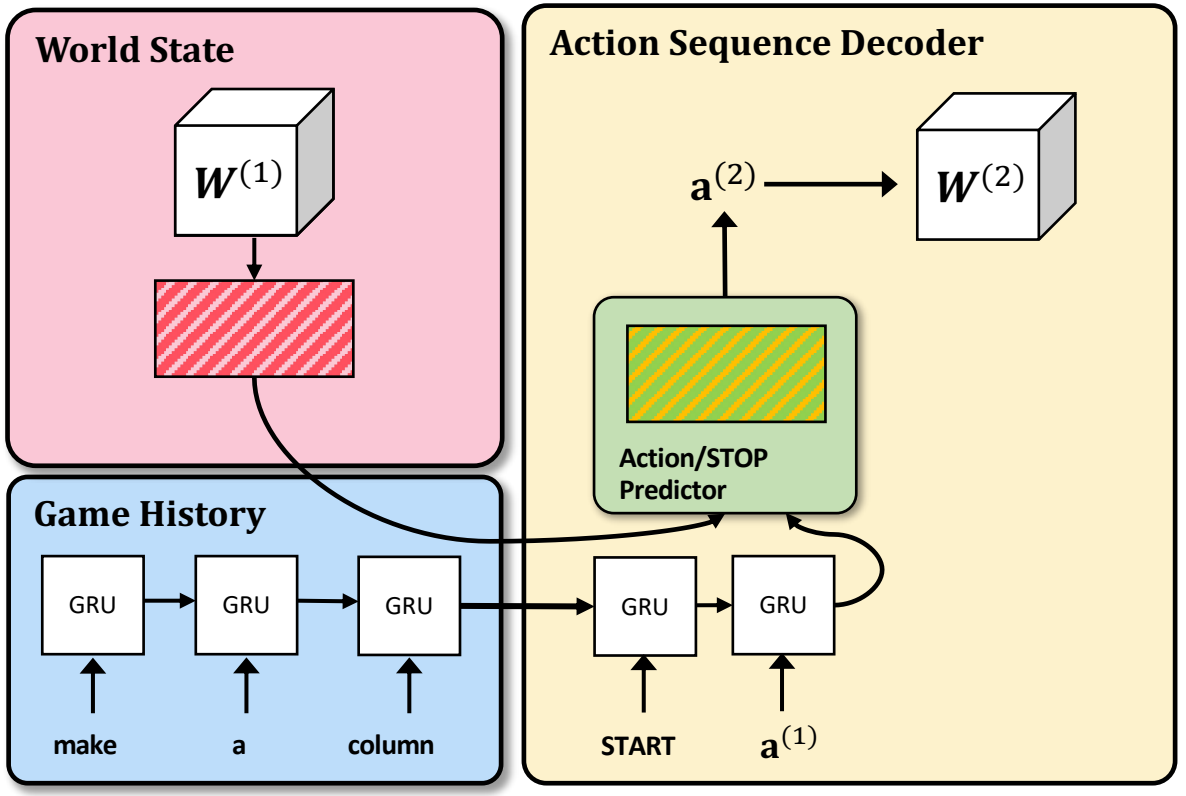
Encoder-decoder network with GRU backbone

Inputs:

- Game history up to $t = 0$
- World state grid $W^{(0)}$

Predicts:

- Sequence of **B** actions $\mathbf{a}^{(0)} \ldots \mathbf{a}^{(t+1)}$ with $\mathbf{a}^{(0)}$ = START



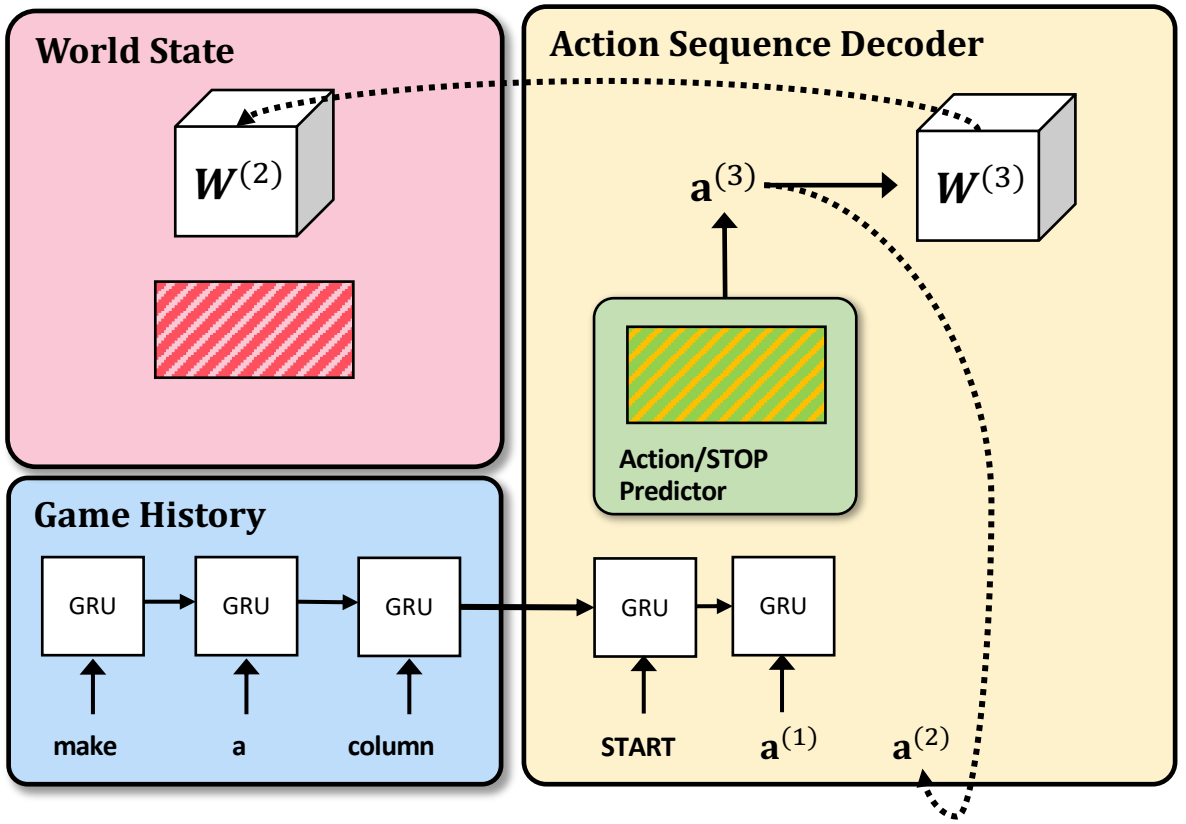
Our Model

Encoder-decoder network with GRU backbone

Inputs:

Game history up to $t = 0$

World state grid $W^{(0)}$



Predicts:

Sequence of $\mathbf{B}$ actions $\mathbf{a}^{(0)} \ldots \mathbf{a}^{(t+1)}$ with $\mathbf{a}^{(0)} = \texttt{START}$

Our Model

Encoder-decoder network with GRU backbone

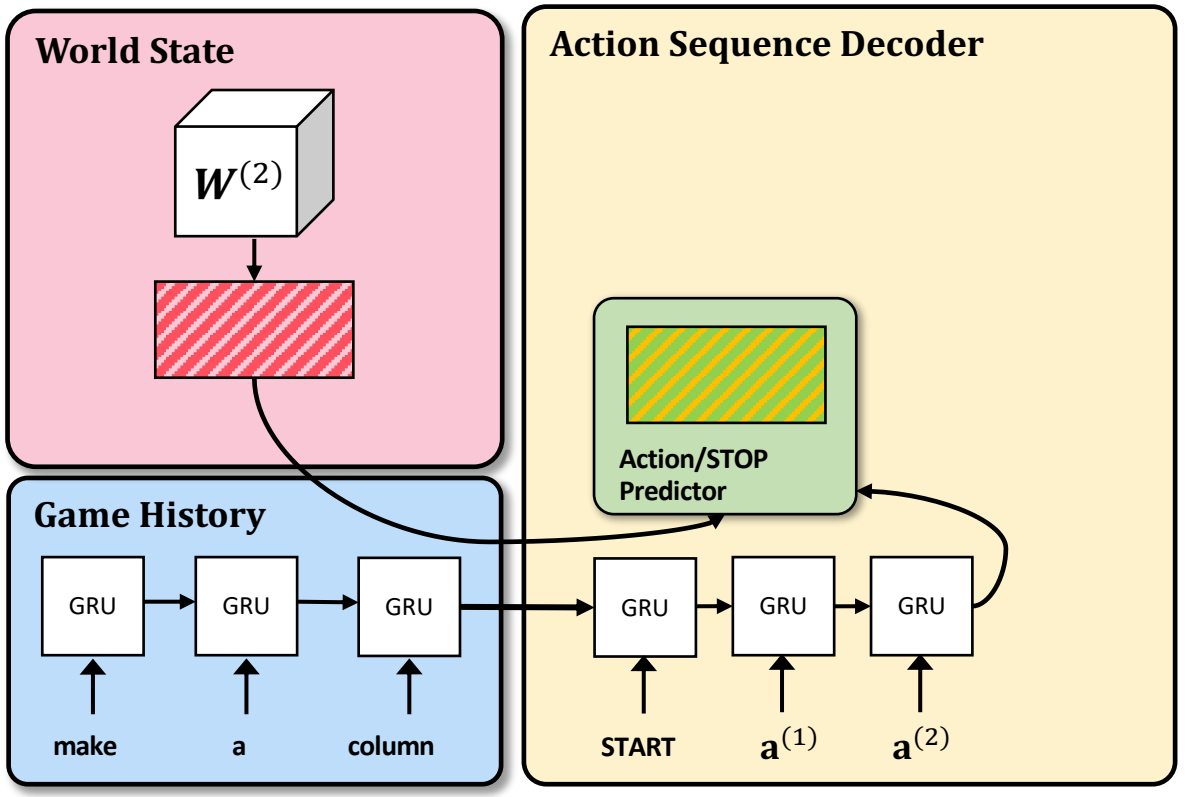
Inputs:

Game history up to $t = 0$

World state grid $W^{(0)}$

Predicts:

Sequence of **B** actions $\mathbf{a}^{(0)} \ldots \mathbf{a}^{(t+1)}$ with $\mathbf{a}^{(0)} = \texttt{START}$



Our Model

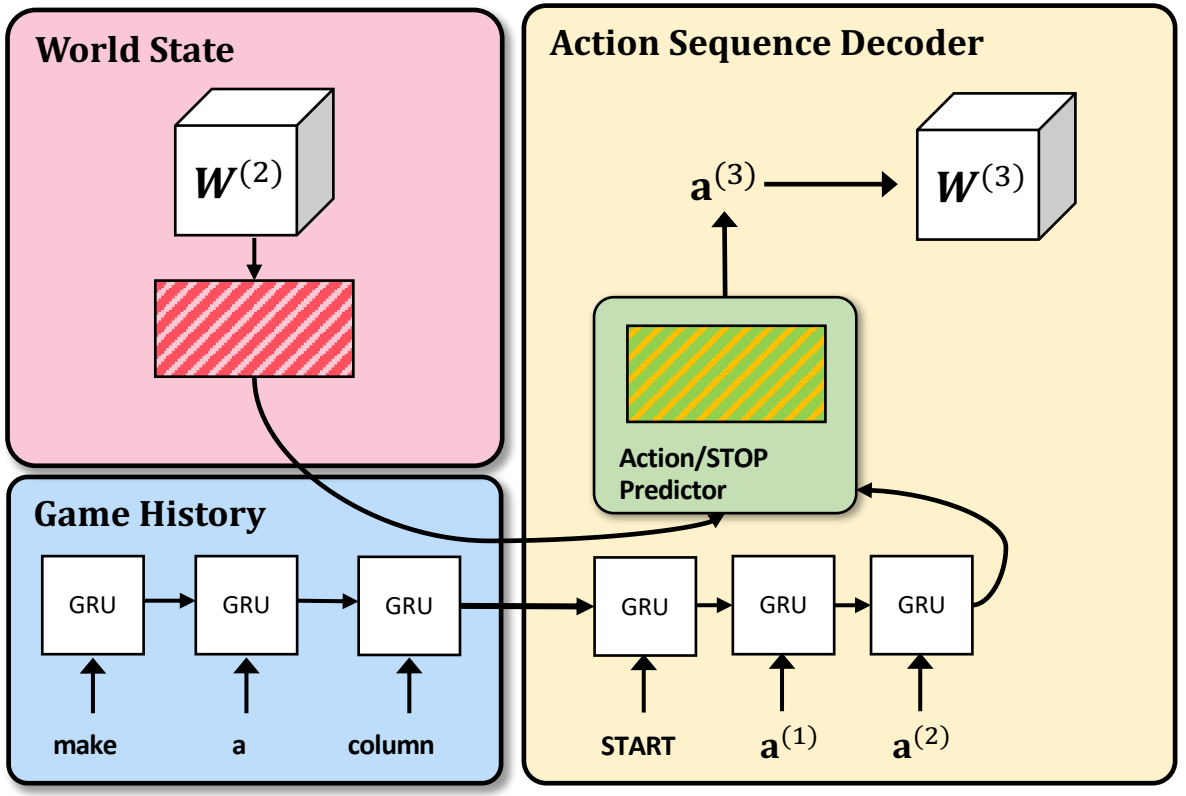
Encoder-decoder network with GRU backbone

Inputs:

- Game history up to $t = 0$
- World state grid $W^{(0)}$

Predicts:

- Sequence of **B** actions $\mathbf{a}^{(0)} \ldots \mathbf{a}^{(t+1)}$ with $\mathbf{a}^{(0)} = \texttt{START}$



Our Model

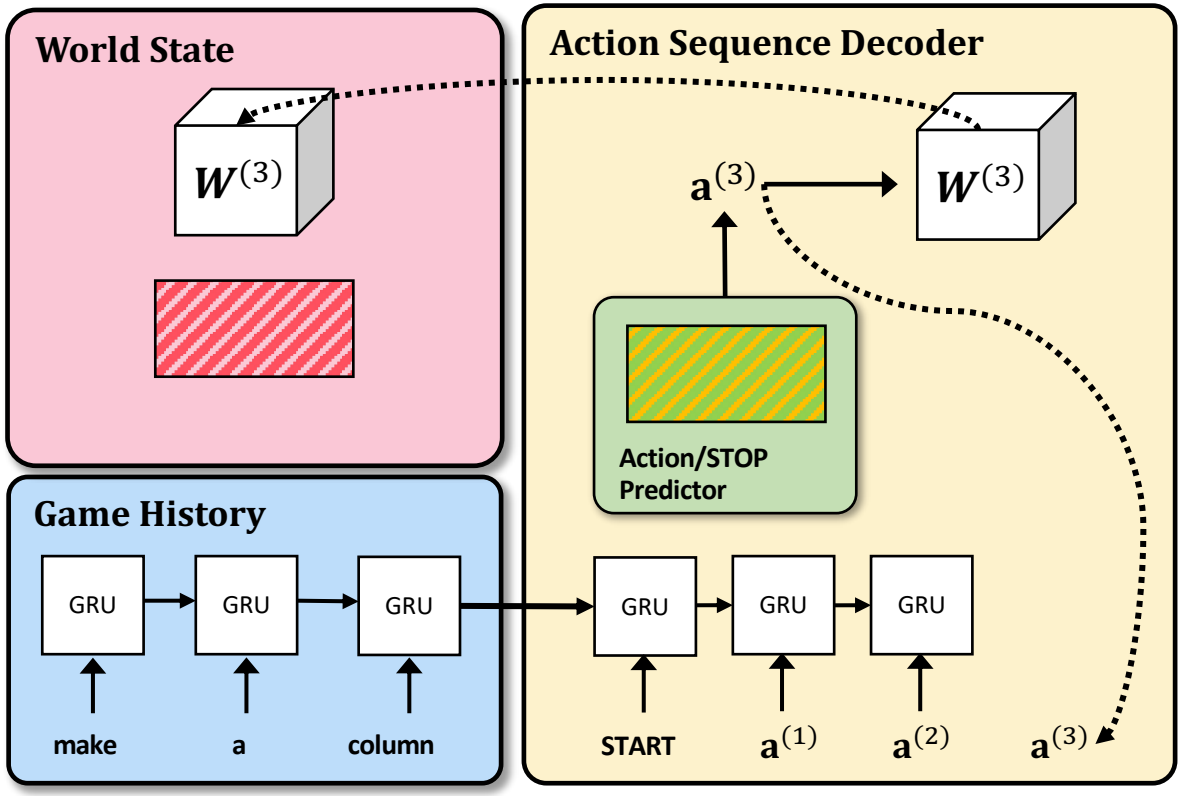
Encoder-decoder network with GRU backbone

Inputs:

- Game history up to $t = 0$
- World state grid $W^{(0)}$

Predicts:

- Sequence of **B** actions $\mathbf{a}^{(0)} \ldots \mathbf{a}^{(t+1)}$ with $\mathbf{a}^{(0)}$ = START



Our Model

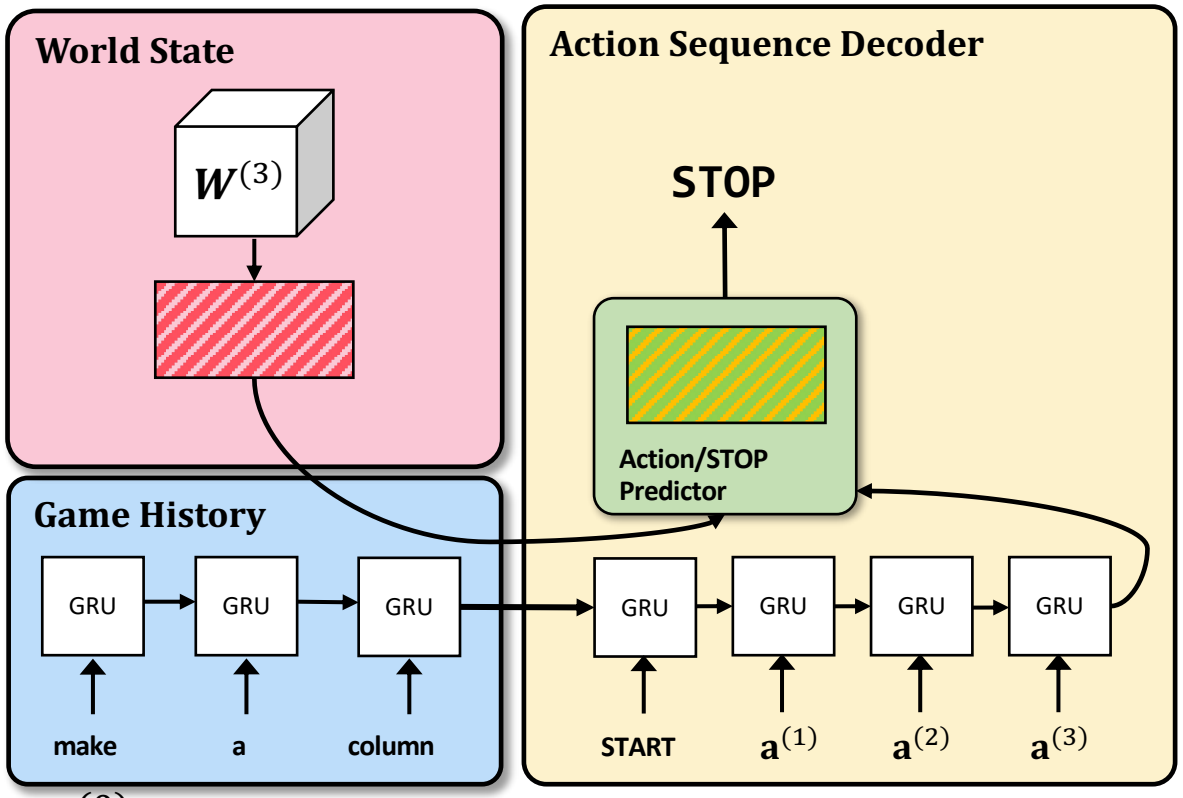
Encoder-decoder network with GRU backbone

Inputs:

- Game history up to $t = 0$
- World state grid $W^{(0)}$

Predicts:

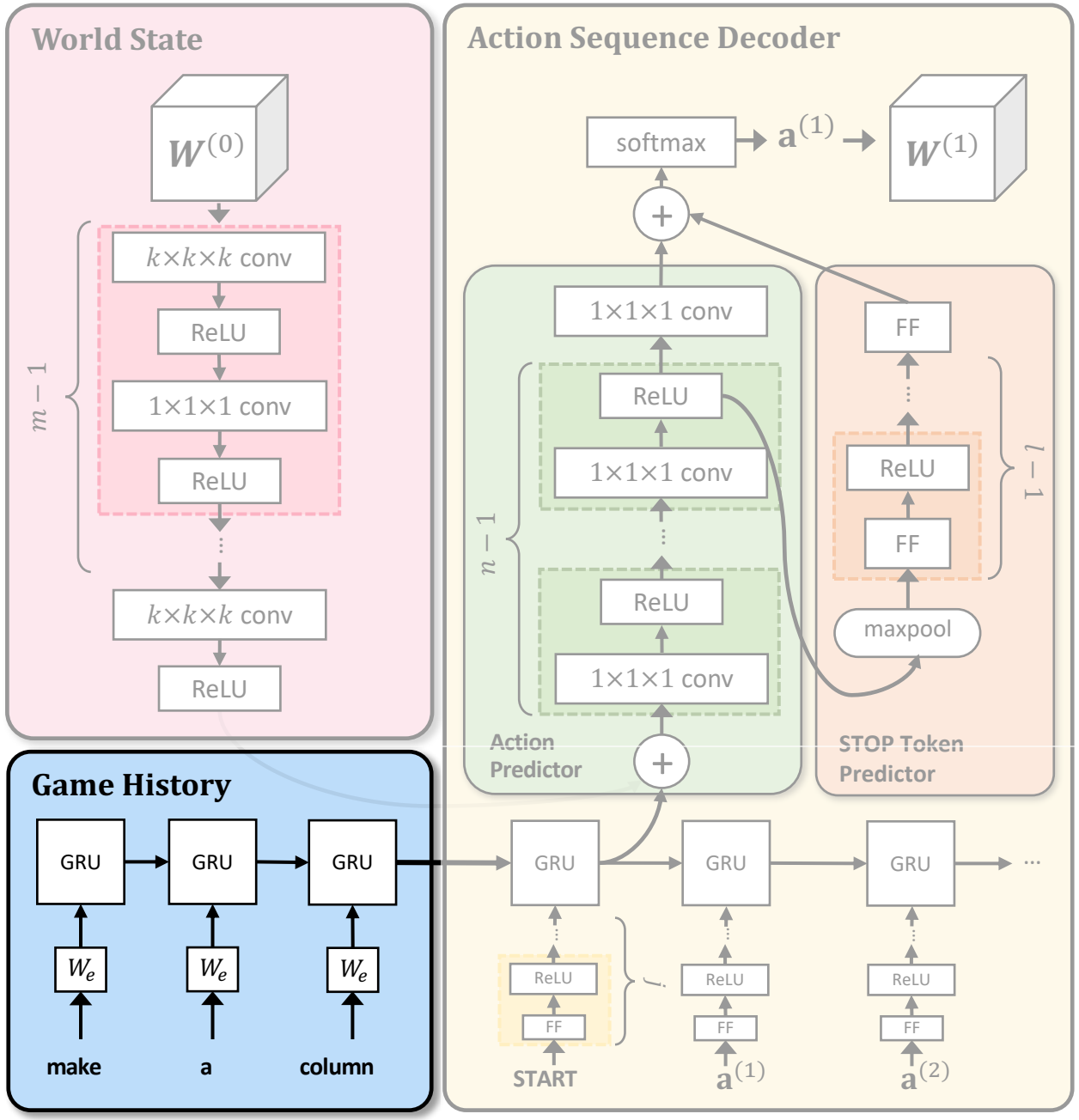
- Sequence of **B** actions $\mathbf{a}^{(0)} \ldots \mathbf{a}^{(t+1)}$ with $\mathbf{a}^{(0)}$ = START and $\mathbf{a}^{(t+1)}$ = STOP



Game History GRU Encoder

Encodes game history
as flat sequence of tokens

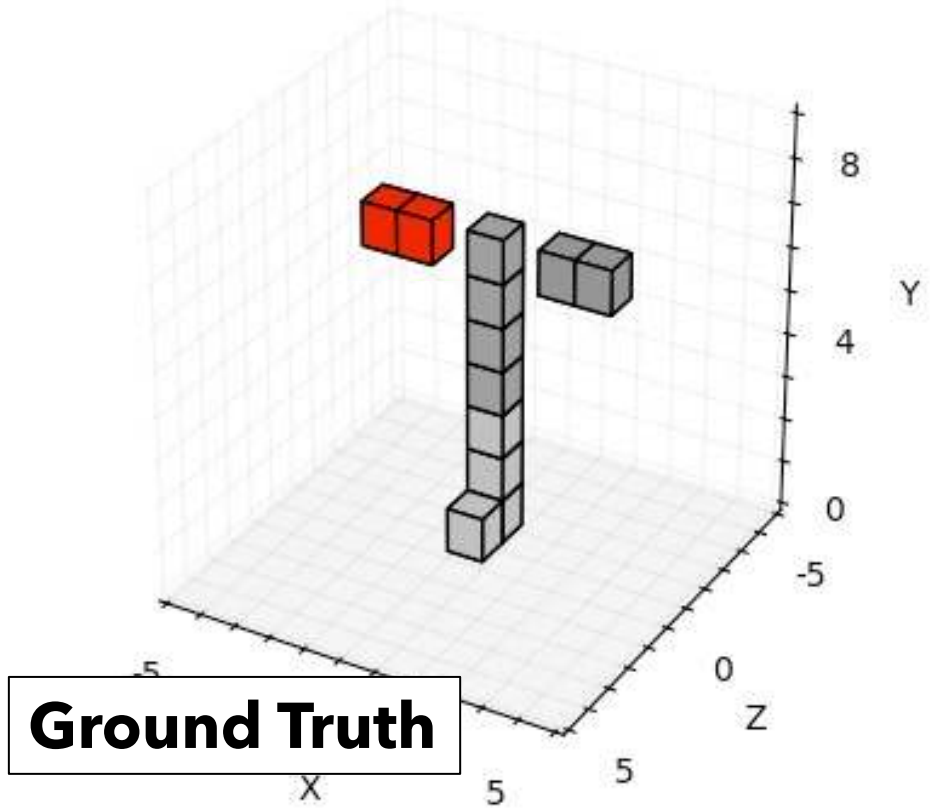
<A> place a red block on the ground </A> <B> like this ? </B>



Encoding Game History: Scheme H1

A's last utterance
and any following **B** utterances

H1
<A> *and the same thing on the other side*



Ground Truth

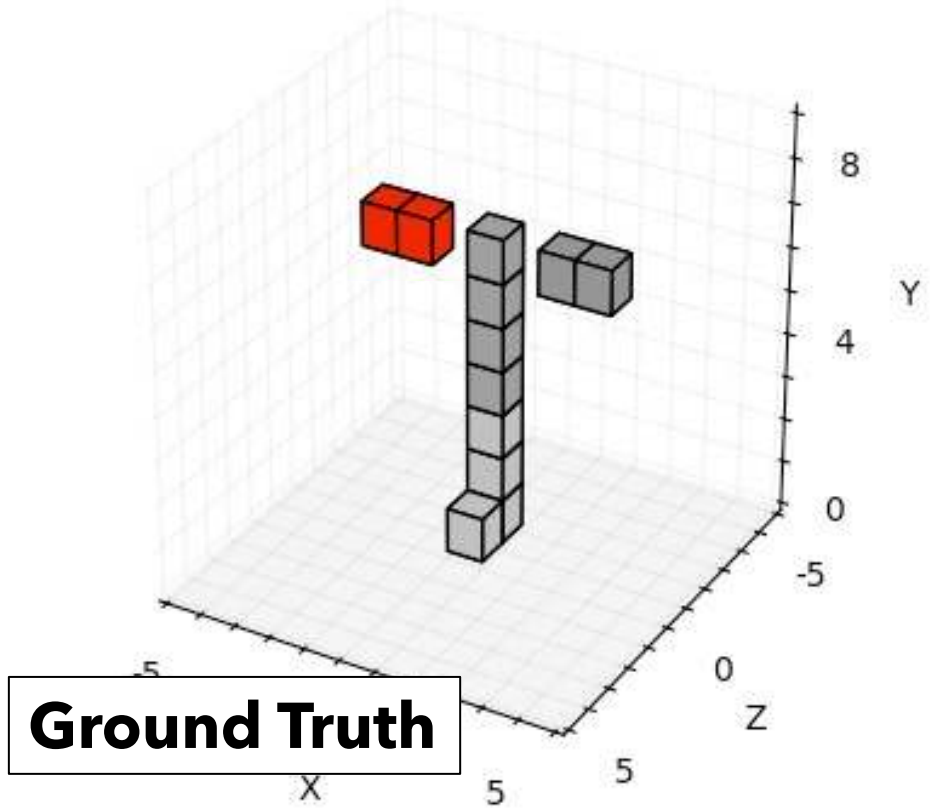
Encoding Game History: Scheme H2

All utterances after
B's penultimate action sequence

H2

<A> *on the same plane facing you, leave a space and then put 2 red blocks down in a row*

<A> *and the same thing on the other side*



Ground Truth

Encoding Game History: Scheme H3

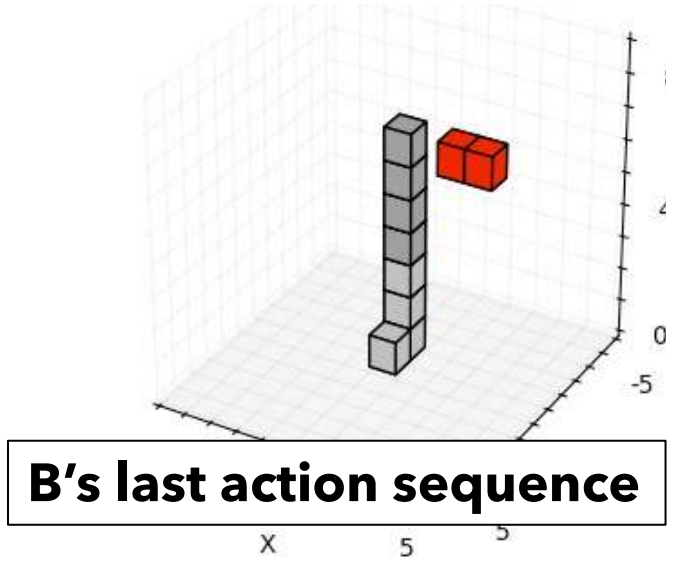
H2 interleaved with
a token representation
of **B**'s last action sequence

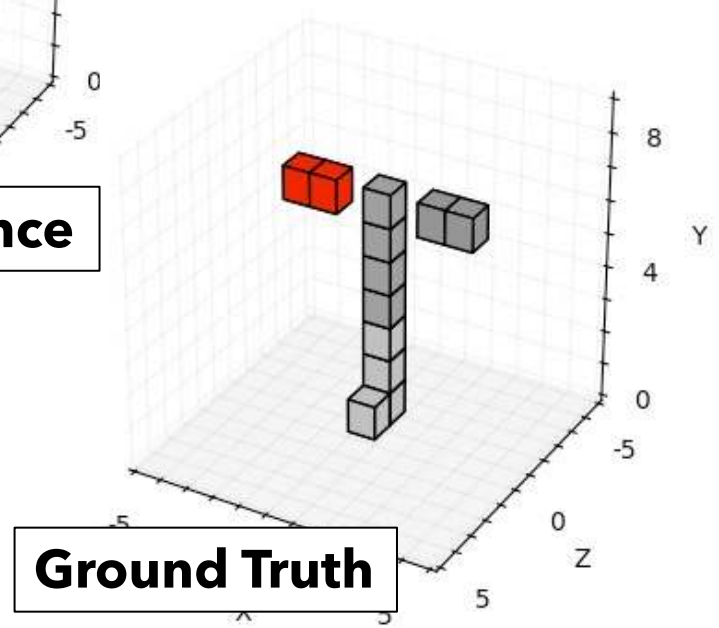
H3

<A> *on the same plane facing you, leave a space and then put 2 red blocks down in a row*

```
<builder_putdown_red>
<builder_putdown_red>
<builder_putdown_red>
<builder_pickup_red>
```

<A> *and the same thing on the other side*



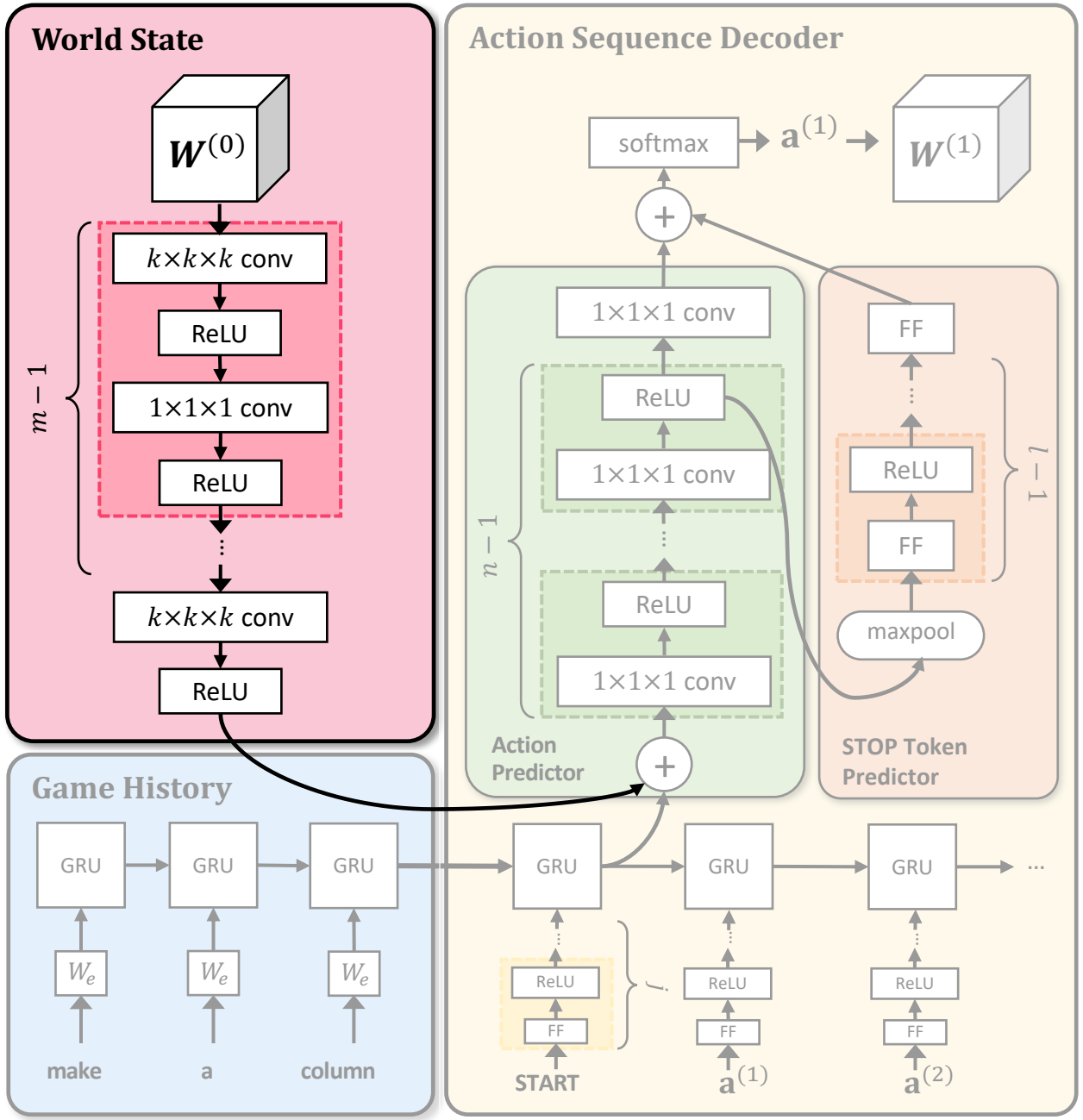


World State CNN Encoder

Encodes the world state at each time step

Input: 11×9×11 3D grid
Each grid cell is represented as a 7-dim 1-hot vector of its block color (or empty)

R	B	G	O	Y	P	Ø



Encoding Action History

Actions that follow each other often affect **adjacent grid cells**

Action history weights

Concatenate an **action history weight α ∈ {0,1,2,3,4,5}**
to each cell's vector representation

R	B	G	O	Y	P	Ø	α

The last five actions get weights from 1 through 5 (least to most recent)

All other actions are weighted 0

Encoding the Builder's Perspective

Spatial relations in instructions (e.g. *"left"*)
often depend on **B**'s perspective (current position and orientation)

Encode **B**'s perspective using **perspective coordinates**

Given: a cell c and the absolute coordinates of the cell $\langle x_c, y_c, z_c \rangle$

Compute relative coordinates of the cell $\langle x'_c, y'_c, z'_c \rangle$
wrt **B**'s current position $\langle x_B, y_B, z_B \rangle$ and orientation (pitch and yaw angles)

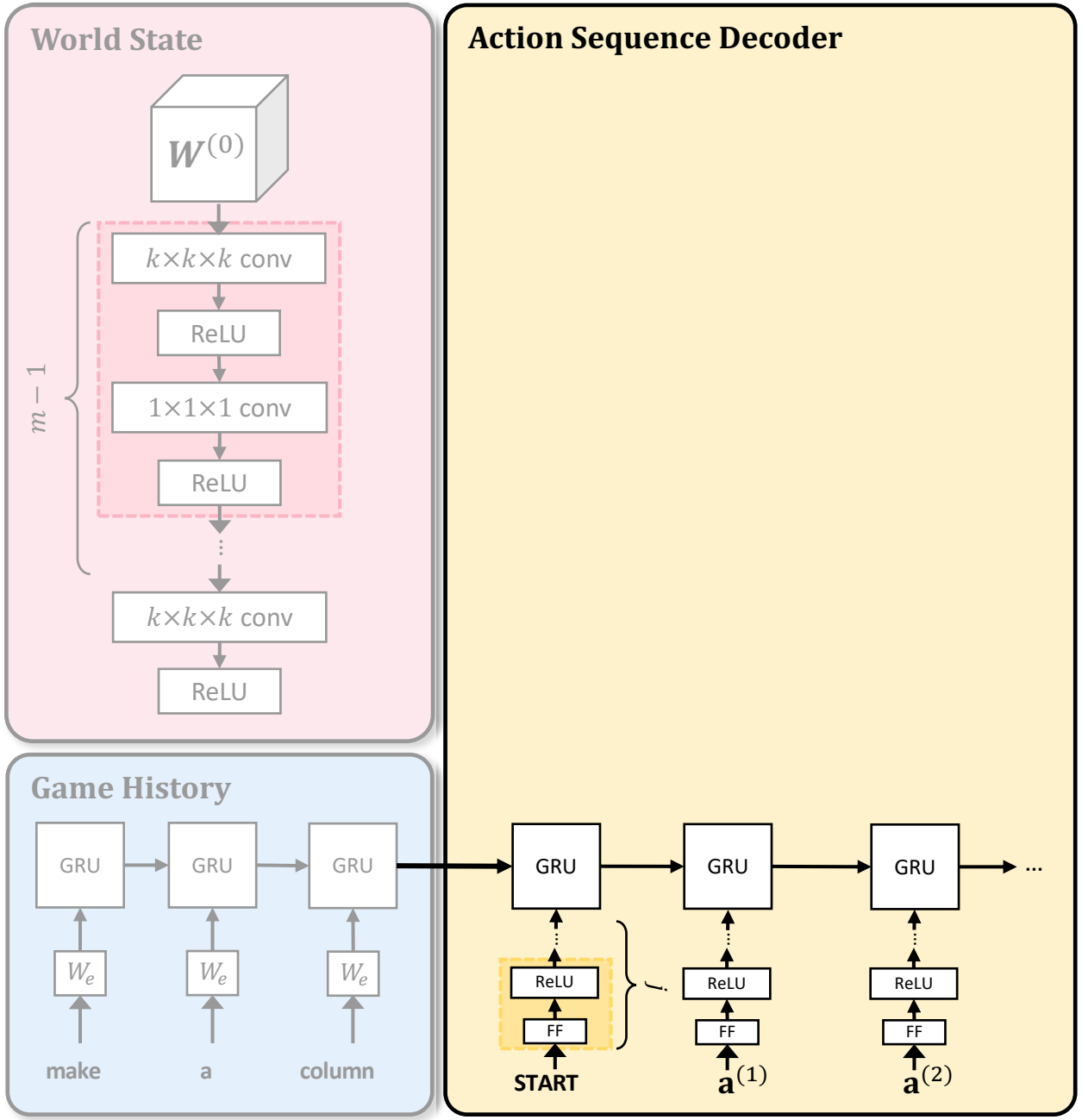
R	B	G	O	Y	P	Ø	α	x'_c	y'_c	z'_c

Action Sequence Decoder

GRU backbone input:

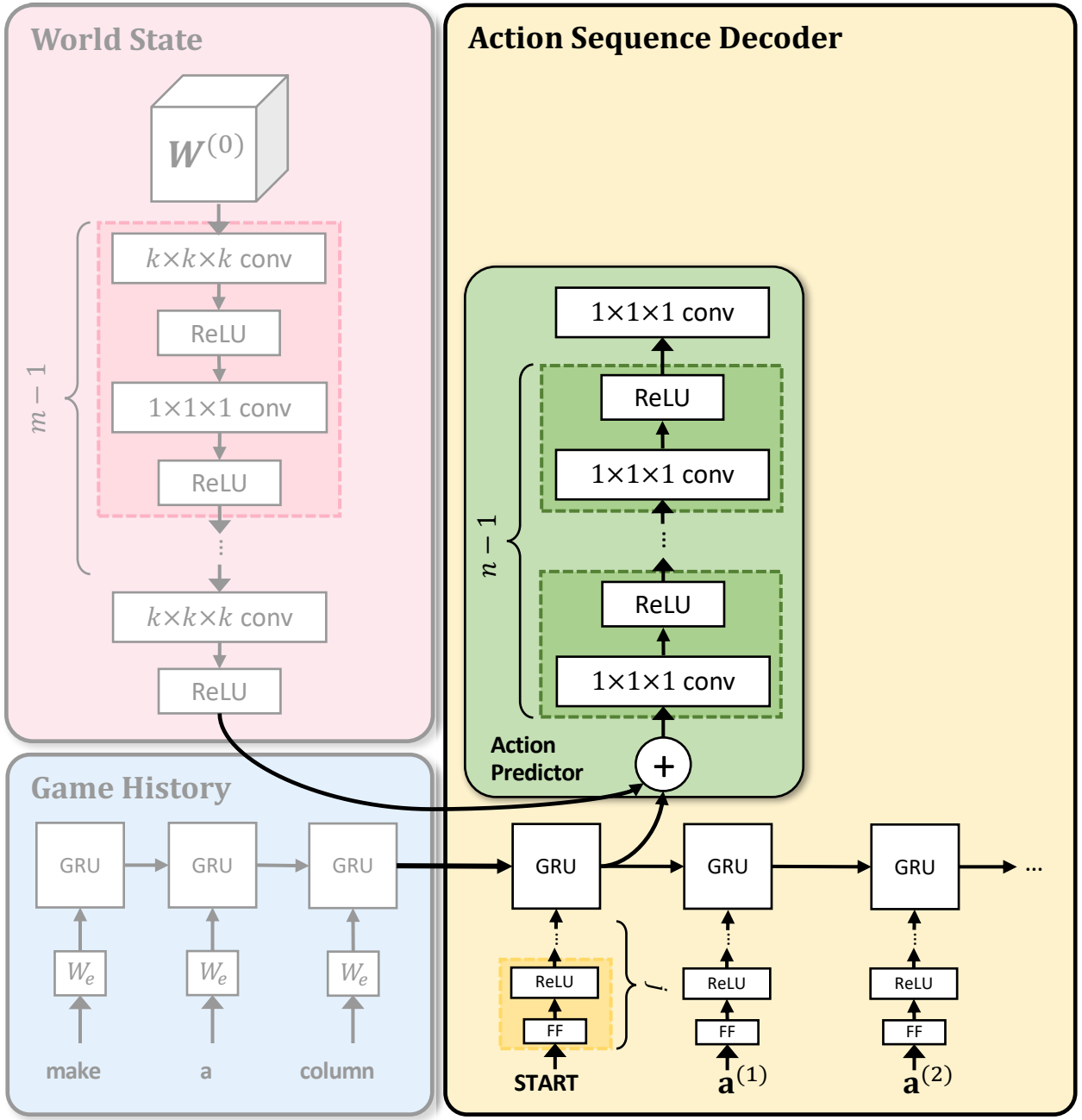
11-dim vector **a** representing action taken at last timestep

Action type		Block color (all 0s if removal)						Absolute coordinates		
p	**r**	**R**	**B**	**G**	**O**	**Y**	**P**	x	y	z



Action Sequence Decoder

CNN action predictor



Action Sequence Decoder

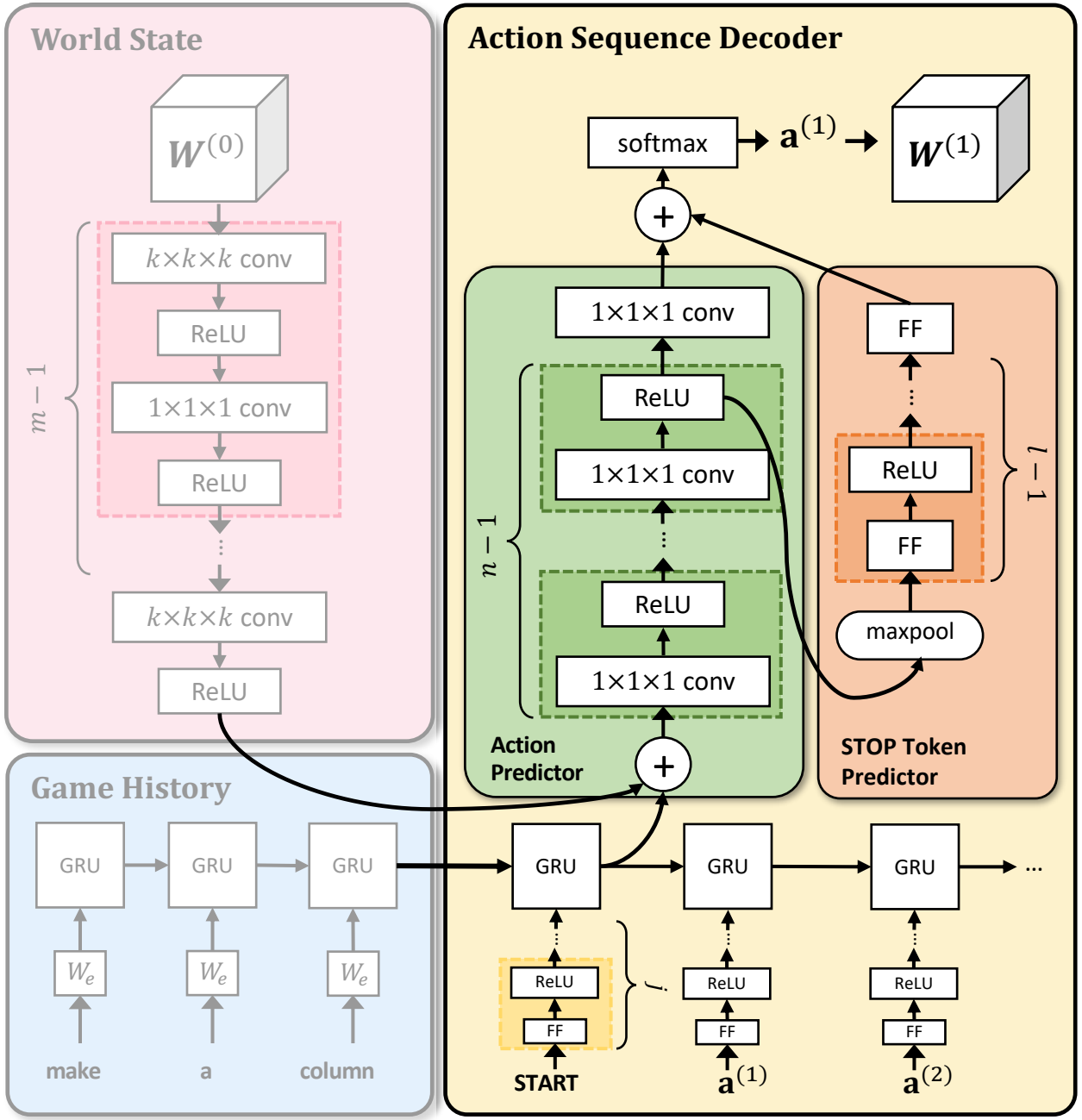
CNN action predictor

STOP token predictor:

- Conditioned on action predictor representation
- Predicts the likelihood of ending the action sequence

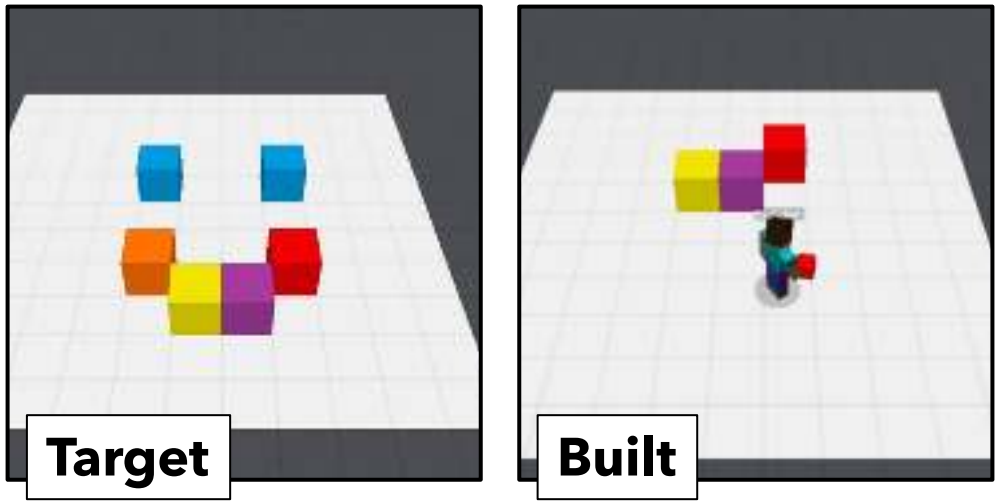
Final prediction:

- Distribution over all possible actions in the grid + STOP probability



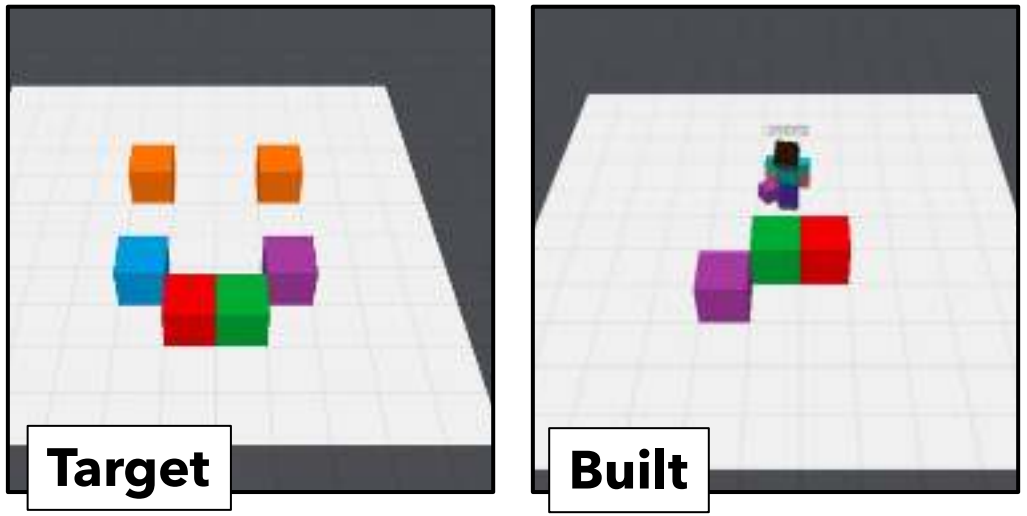
Data Augmentation

Original



<A> now take a red block
<A> place it in the square diagonally to the right of the purple block
<A> nice
<B> thank you
<A> now do the same thing on the other side but with an orange block
<B> other side of the purple or yellow?
<A> yellow

Augmented



<A> now take a purple brick
<A> place it in the square diagonally to the right of the green block
<A> alright
<B> thank you
<A> now do the same thing on the other side however with an blue block
<B> other side of the green or red?
<A> red

Evaluation: Net Actions F1

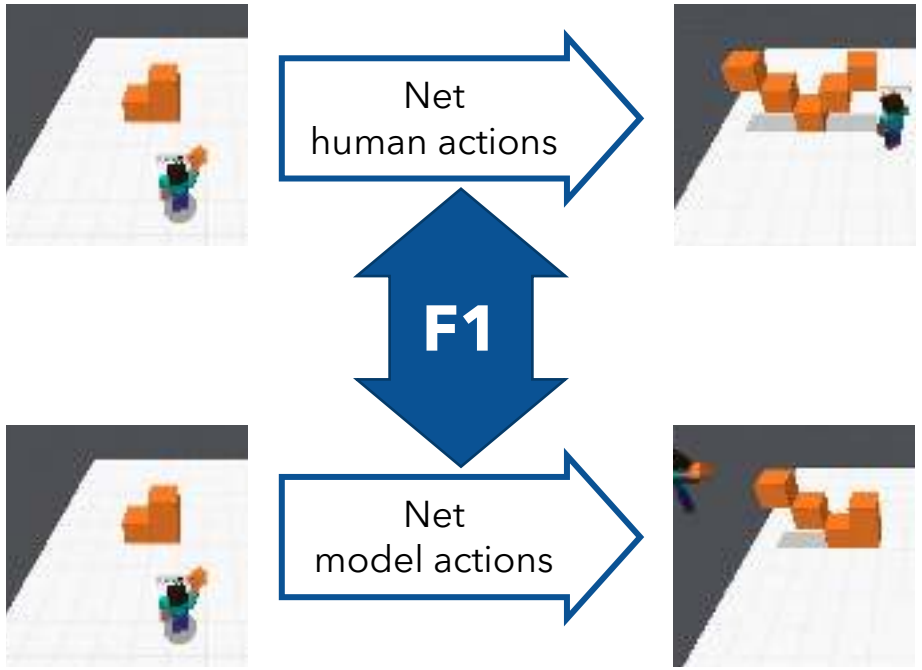
Net actions ignore the order of actions
and blocks that were placed and removed in the same sequence

Evaluation: Net Actions F1

Net actions ignore the order of actions
and blocks that were placed and removed in the same sequence

Evaluation: Net Actions F1

We compute a **micro-averaged F1** between net actions in the **ground truth (human)** sequence A_h and in the model's **predicted** sequence A_m



Quantitative Results

3,709 train / 1,616 test / 1,331 dev **B** action sequences
(splits across target structures)

Supervised training to minimize cross entropy loss; greedy decoding

	H1	H2	H3
BAP-base	11.8	12.4	**14.6**

Richer game history helps increase performance

Quantitative Results

3,709 train / 1,616 test / 1,331 dev **B** action sequences
(splits across target structures)

Supervised training to minimize cross entropy loss; greedy decoding

	H1	H2	H3
BAP-base	11.8	12.4	14.6
+ action history	14.6	18.2	**19.7**

Richer world state representations help increase performance

Quantitative Results

3,709 train / 1,616 test / 1,331 dev **B** action sequences
(splits across target structures)

Supervised training to minimize cross entropy loss; greedy decoding

	H1	H2	H3
BAP-base	11.8	12.4	14.6
+ action history	14.6	18.2	**19.7**
+ perspective	15.7	18.7	18.8

Richer world state representations help increase performance

Quantitative Results

3,709 train / 1,616 test / 1,331 dev **B** action sequences
(splits across target structures)

Only 21.2 F1? That's pretty bad, right?

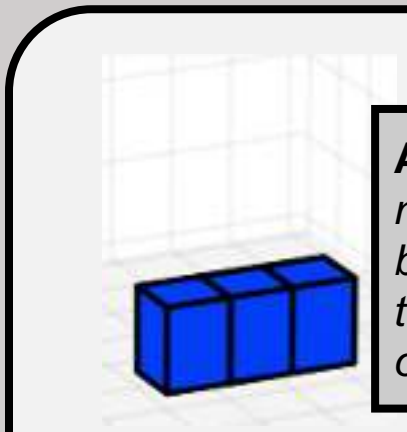
Supervised training to minimize cross entropy loss; greedy decoding

	Trained on original data			On augmented dataset		
	H1	**H2**	**H3**	**H3 + 2x**	**H3 + 4x**	**H3 + 6x**
BAP-base	11.8	12.4	14.6	15.6	16.1	17.0
+ action history	14.6	18.2	**19.7**	16.9	20.0	18.4
+ perspective	15.7	18.7	18.8	19.5	**21.2**	20.8

Data augmentation helps increase performance.
Now we get the best results with the full world state representation.

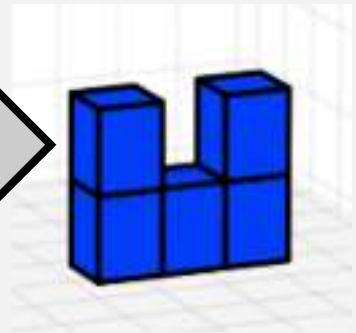
What can the Neural Builder do?

Perfect interpretation of
*"do **it** one more time"*
*("**it**" = "place two blue blocks on top of the edges of the line")*



Penultimate Built State

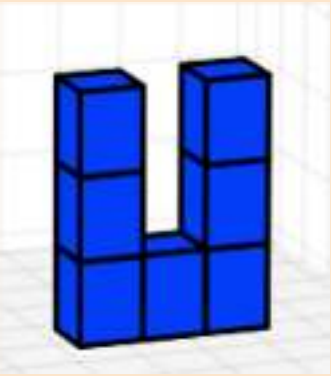
Architect: *now place two blue blocks on top of the edges of the line*



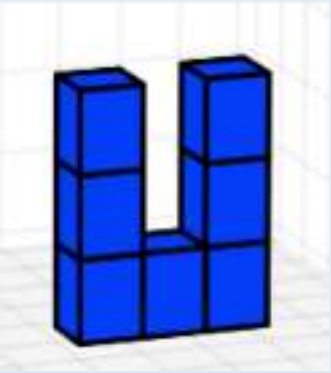
Last Built State

Human-Human Game History

Architect: *do it one more time*



Builder's Actions

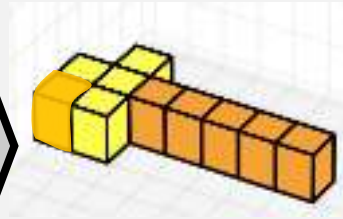


Builder's Actions

What can the Neural Builder do?

Plausible interpretation of
*"and do **the same** on **the other side**"*
***"the same"**="the next two blocks"* ***"the other side"**=???*

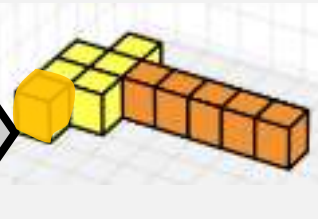
Architect:
the next two blocks will be off the corners of each of those, in the direction of the last yellow block.



Penultimate Built State

Builder:
like that, or somewhere else?

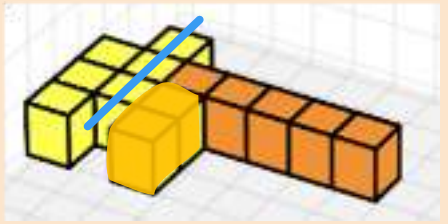
Architect:
add one more block to the end of that on your side



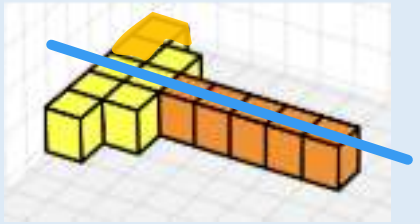
Last Built State

Human-Human Game History

Architect:
and do the same on the other side



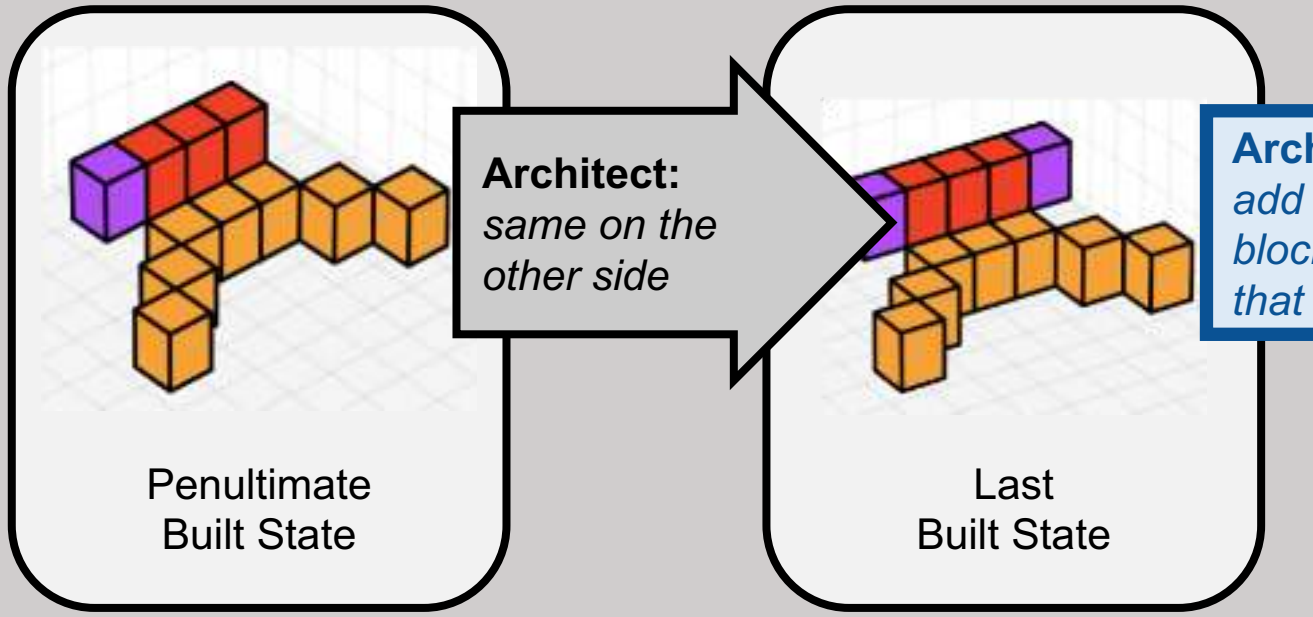
Neural Builder's Actions

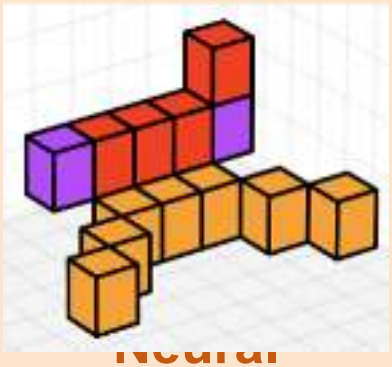


Human Builder's Actions

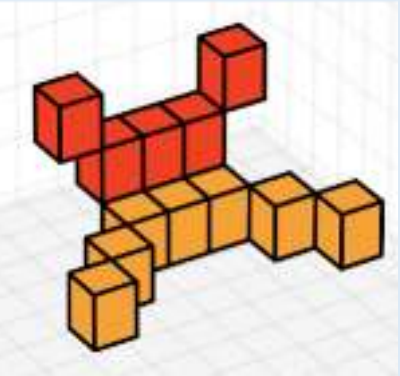
What can the Neural Builder do?

Correct interpretation of
"add one red block on top of that"
.... but the human anticipated the next steps





Neural Builder's Actions



Builder's Actions

Plugging the Builder into Minecraft (floating blocks)

Plugging the Builder into Minecraft (Spatial relations; "The Gap")

Where Do We Go From Here?

What Have We Accomplished?

Supervised training on relatively small amounts of data with simple end-to-end neural models and no linguistic annotation yields (surprisingly?) **decent baseline models:**

– **The Architect gives block-by-block instructions** that are **fluent** and often (but far from always) **correct**

– **The Builder executes instructions** in ways that are often **correct** or **plausible**

– **The Builder shows some understanding of complex concepts and context** (row, middle, gap, the same, other side)

What Remains To Be Done?

We haven't yet *solved* the tasks we started working on

– We need **higher accuracy** of instructions and executions

– We want the Architect to generate **richer, more diverse utterances**

This requires **richer models**, possibly **more data,**
and other **training regimes**

– What's the role of **explicit domain knowledge?**

– Naively using **3D CNNs** as world state representations for the architect doesn't seem to work, because there is not enough supervision.

What Remains To Be Done?

Fully interactive agents require further capabilities:

– Both systems need to be trained for **task completion**

– **The Builder needs to speak**, but this requires knowing **what to ask**

– Both agents need to know ***when*** **to speak**

Thank You!

http://juliahmr.cs.illinois.edu/Minecraft

juliahmr@illinois.edu

This work was supported by Contract W911NF-15-1-0461 with the US Defense Advanced Research Projects Agency (DARPA) Communicating with Computers Program and the Army Research Office (ARO).