Move a Box: A Decentralized Online 3D Game

Peng Wang (`pwang1`); Di Zhu (`dzhu1`)

1 Introduction

Everyone played LEGO in their childhood. Now we've brought this experience into the cyber world.

Move a Box (fig 1) is a 3D sandbox game where you pile up building blocks to create constructions. There's real physics to bring you the most realistic experience, and you're able to play it over the Internet with many of your friends.

In the world of *Move a Box* there're infinite possibilities: building a magnificent edifice to impress your peers, filming a 3D animation of your own (game) world out of imagination, or simply "kicking" a soccer ball with your friends ... The list is endless!

Figure 1: In-game snapshot: watching another player who's carrying a piece of brick

On the technical side, the game logic of *Move a Box* is written with Unity 5 and C#. The multi-player part heavily relies on a robust network layer implemented with Go. The game controller makes use of a network manager (`netman`) to exchange real-time status with other peers in the same game. `netman` utilizes the LSP network

protocol for lightweight and reliable message transfer, as well as fast failure detection. Because there's no centralized server, it also uses the Paxos algorithm to ensure consistency of game status among all players.

2 Get it Running

2.1 Binaries and System Requirements

We've prepared compiled binary files of the game for Windows and macOS, which can be downloaded from `https://www.dropbox.com/sh/l5biokz4wuv02ij/AACQhNc_9Uv6g-L_T_bQ4PUha`. They are supposed to be working out-of-box, meaning all you need to do is unzipping and executing `MoveABox.exe` (Windows) or `MoveABox.app` (macOS). The system requirements are [1]:

- OS: Windows XP SP2+ or Mac OS X 10.8+.
- Graphics card: DX9 (shader model 3.0) or DX11 (feature level 9.3 capabilities).
- CPU: with SSE2 instruction set.
- Network connection: required for multi-player mode.

It has been tested on Windows 8.1 and macOS 10.12.

** On macOS, you might see "MoveABox can't be opened because it is from an unidentified developer." This is imposed by Apple [2]. Please hold the Control key, then click the app icon, then choose Open from the shortcut menu. Click Open.*

2.2 Building from Source

If you are running a different OS, or it's simply the right thing to do, it's very easy to build the game from source code. Please see Unity's website [1] and Go's website [3] for system requirements for editors / compilers.

Assuming you have Go properly installed and configured, we'll first compile `netman` as an executable file. Please set environment variable `GOPATH` to the directory which contains the `src` folder of the source code. Let's `go build` the `netman` package.

On Windows, say `src` is located in `C:\MoveABox`, type in a `cmd` window:

```
set GOPATH=C:\MoveABox
go build github.com\cmu440-F16\paxosapp\netman
```

... and you'll get a `netman.exe`.

On macOS or Linux, suppose `src` is located under `/home/Judy/go`, execute:

[1] `https://unity3d.com/unity/system-requirements`
[2] `https://support.apple.com/kb/ph18657`
[3] `https://golang.org/doc/install#requirements`

```
export GOPATH=/home/Judy/go
go build github.com/cmu440-F16/paxosapp/netman
```

... and a `netman` binary is built.

Next, we'll build the game itself. It's strongly advised to open the project in Unity with version $\geq$ 5.5.0f3. Please open the main scene (`Main.unity`) in Unity Editor, then click `File - Build Settings ...` . Here you're able to choose target platform and architecture. Click on `build` to make the executable.

Finally, put the generated `MoveABox` in the same folder as `netman`. Ready to go!

3 Explore the Game World

3.1 Game Overview

What are the rules and targets of the game? Nothing! Think it as a virtualized LEGO experience, and with realistic physics. As mentioned above, you are absolutely free to move around in the game world. Press arrow keys or W/A/S/D to walk in four directions, and hit space bar to jump. You can configure all buttons at start-up.

You will notice a small circle in the center of the screen (see fig 2). It basically

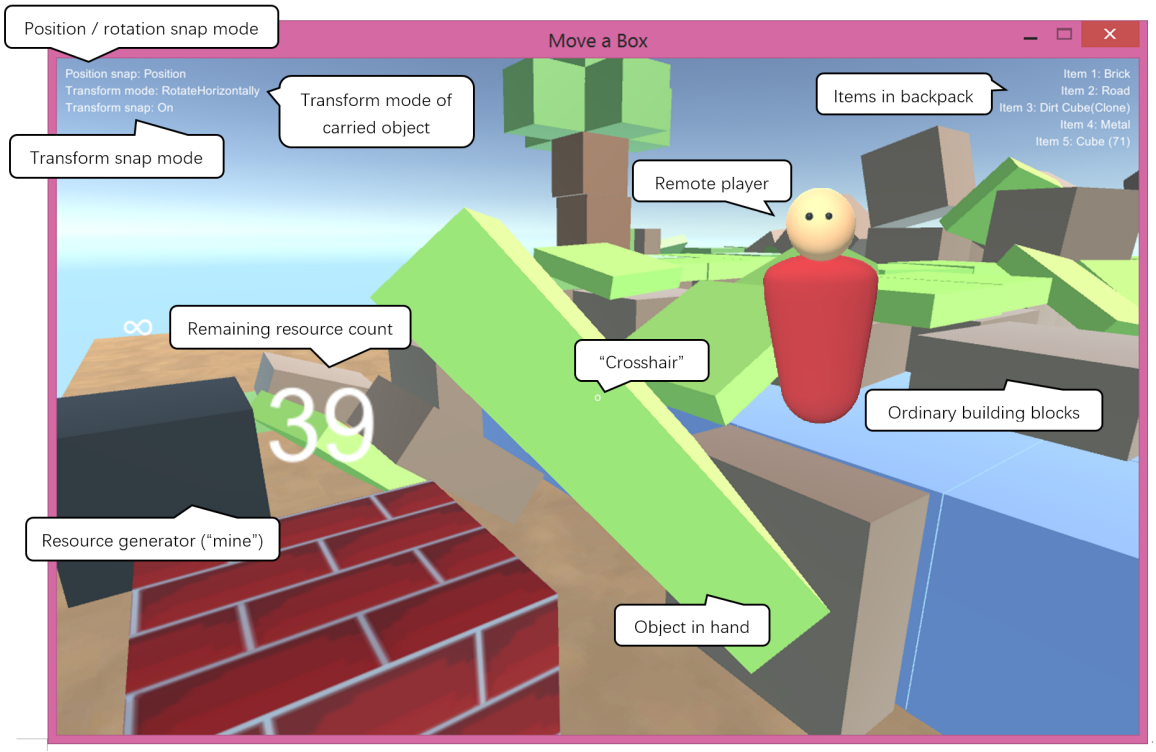

Figure 2: Components in the game view

acts as a crosshair and indicates which object you are pointing to. If you feel like picking up something, approach and aim at it, then press left mouse button (again, you can change that in input settings). Now you may "carry" it around, and press left mouse button again to drop it.

Many items in the game scene can be picked up, and there's another special sort of objects called *resource generators*. The resemble "mines" in the real world, in that you can take out some amount of specific items out of it. All these entities have numbers above them, indicating how many items are left (1 to ∞). Plan wise with scarce resources (especially in multi-player games)!

It's also possible to add a bit variety. When holding an object, the top-left corner of the screen (fig 2) shows current *transform mode*. Press enter key (or push down mouse wheel) to switch among different modes, and use minus (-) and plus (+) keys (or scroll mouse wheel) to adjust certain properties. For example, some objects rotates horizontally or vertically, and scales along all axes, while some others don't. (Why will anyone want to rotate a perfect ball?)

Finally, here's a good piece of news for perfectionists. It's always hard to align things perfectly when piling up boxes in life, let alone in game. That's why we provide a position / snap mode (indicated on the top-left corner). If it's on, whenever you drop an object, it will automatically align to a grid point and / or parallel to X / Y axis. The default key for switching is "[". Similar snap options for *transform mode* is also available by pressing "]". Try it and you'll thank me for that.

At any time, press tilde key (~) to pause or quit game.

3.2 Single Player Mode

When the game is started, you may choose between single- or multi-player mode. Let's first try it locally. Single-player mode enjoys all the features in the last section, and here are a few more:

We know it can be tiring to run to and fro between "mines" and "construction sites". That's why we introduce the *backpack*. When you pick up an object, press right mouse button (as always, it's configurable) to put that item in your backpack. Your inventory is displayed on the top-right corner of the screen (fig 2). To bring out an item from the backpack, simply press the corresponding number key.

Let's say you've spent 10 days and built a Taj Mahal with bricks. Very impressive. But what if you want to move it a little to the left? *Grouping* to the rescue! Press TAB to switch to bird's-eye view, then (move with arrow keys and zoom with -/+ and) use your mouse to select a range. All objects in this range will be marked with a yellow circle (fig 3). Hit enter to group them. Now you are ready to carry the whole

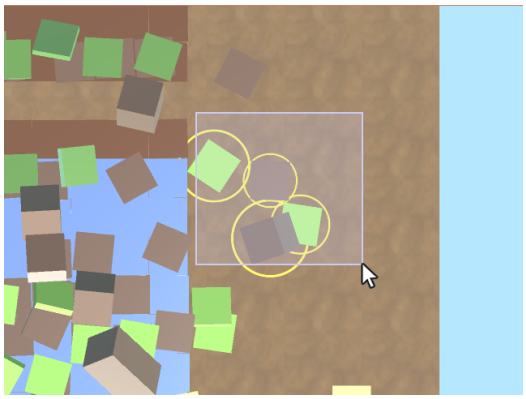

Figure 3: Selection in bird's-view

Figure 4: Multi-player connection

object at once. At destination, do the same things to ungroup them.

Last but not least, you will find save / load options in the pause menu.

3.3 Multiple Player Mode

Time to start an online game! To get everyone connected, one player needs to click the "Let others input my address" button. His IP address [4] and port will be displayed below (fig 4). Everyone else puts that address in the text box and hit "connect". Now the list of all connected players are shown, and they wait for the first person to start the game.

Note there isn't really a "server" for the game. Once the actual game starts, everyone is playing the same role in the network. If one person happens to get disconnected from the Internet, the rest can still play smoothly. However, when you lose half of members, the game has to stop and you'll be prompted.

In this mode, sometimes players will fight over their favorite objects. Everyone is free to pick up any unattended items, but there can be at most one person holding the same item. Don't worry! This is handled by our program and whoever clicks the mouse earlier gets ownership. Remember the other person can still take it away after you drop it.

We will leave it to you to invent many new and interesting ways to play it.

[4] Things can get complicated if players are in different networks, e.g. one is behind a Wi-Fi router and the other is using a VPN. Try connecting to the same LAN or all having public IP addresses.

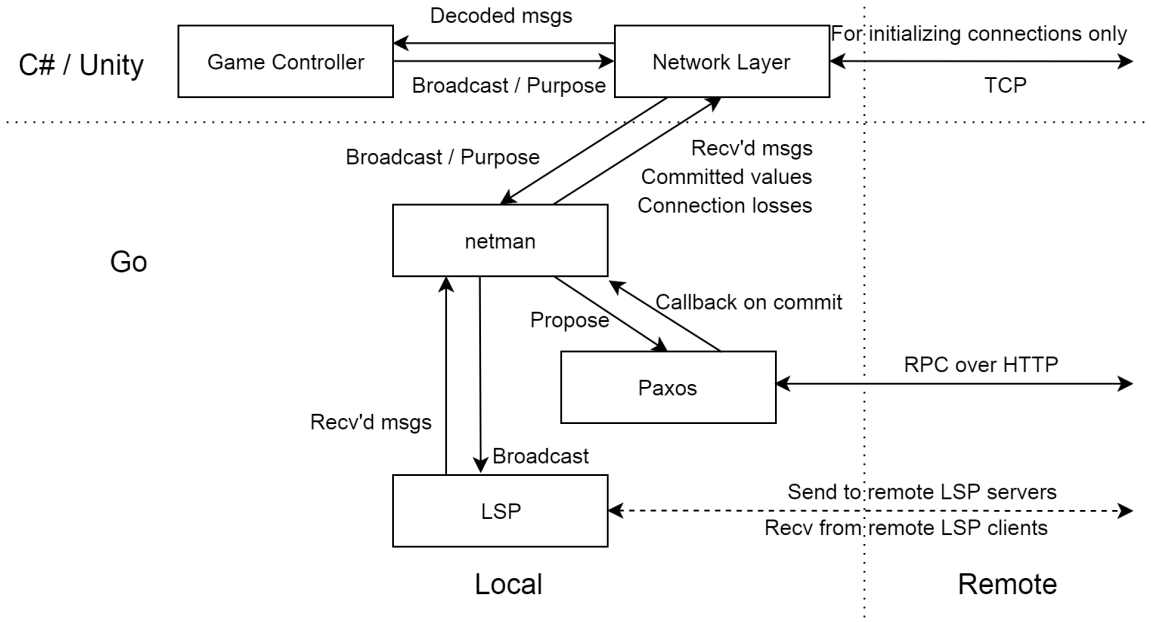

Figure 5: System architecture of network components of the game

4 Technical Details: Network

4.1 System Architecture

Figure 5 generalizes the whole structure of components that support the multi-player mode of *Move a Box*.

The `Game Controller` handles the flow and main logics of the game. It communicates with all remote machines through `Network Layer`, which temporarily uses TCP to set up connections among all players at the beginning, and then exchange messages solely using `netman` throughout the entire game.

`netman` is written in Go and manages network resources at a low level. On every machine, it sets up two different network services to satisfy the different needs of the application. For data entries that requires consistency among all nodes, the Paxos algorithm is adopted. For other non-conflicting updates, it broadcasts messages through the LSP protocol to LSP servers on remote nodes.

Some important details of these two levels are discussed in the sections below. We omit some ordinary routines and the game logics itself because they are less relevant.

4.2 Low-level Communication

`netman` handles all network messages during the gameplay. It works at a low level in the sense that it has no knowledge about the contents of messages. It only exports

two simple commands: `Broadcast` (sending a message to everyone) and `Propose` (using Paxos ask everyone to agree on the value of a key) and communicates with the application through standard input / output.

At start-up, `netman` receives a list of the network addresses of all nodes, and sets up Paxos as well as LSP server and clients. These modules come from previous projects with only fractional modification.

Paxos is adopted whenever a node determines a value for a specific key, which others may also want to write. This ensures consistency among all players and is required in many scenarios, in the absence of a centralized server. We don't use one because any node may fail during the whole game process. We modified the implementation of Paxos, so that the application is notified as soon as this node receives a commit command. In this way, the application doesn't need to poll on all keys, and it also avoids some otherwise nasty race conditions.

On the other hand, Paxos incurs too much network traffic and takes long to complete. This is unacceptable in real-time games if every update has to go through it. Therefore, we leverage the LSP (Live Sequence Protocol) for message exchange of non-conflicting information, e.g. movement of players. LSP has strengths over TCP in that it's more light-weight yet still reliable. Besides, LSP is able to detect failed nodes very fast, which is crucial to our Paxos implementation: a Paxos propose requires successful commits to all nodes before it returns, so it's necessary to remove disconnected ones in time.

There's a small issue in integrating LSP into this project: LSP distinguishes servers and clients, but peers are considered equal in our model. For simplicity, on each node we run a LSP server and let everybody connect to it.

4.3 Application-level Design

The first thing `Network Layer` does is getting all players connected to each other. We use a server-client model over TCP for this initial setup stage (with heartbeat mechanism etc.), but this is just aimed at helping players connect. As soon as the game starts, the network structure becomes decentralized and relies solely on `netman`. All messages are serialized into string and there isn't much novelty here.

Whenever possible, we prefer the faster Broadcast command over Paxos. For example, a player's current position is only determined by his local instance of program and will not conflict with others. Therefore it's safe to periodically disseminate the character's transform. Others who receive such messages can simply place this figure accordingly. We also use linear interpolation to make this movement look smooth (a node only sends out a couple of position update message per second).

Meanwhile, there are lots of status that all nodes might want to update, and we use Paxos in these cases. Let's think of the following scenario: Every block has an `owner` property, and a player is only allowed to *attempt* to pick block `i` up if `owner(i) == null`. He proposes on key "block(i)" with his player ID as value. If the proposal succeeds with desired value, then he sees him carrying the block locally. On the other hand, anyone receiving a commit message `<key = block(i), value = id>` where `id != my_id` atomically set `owner(i)` to `id`. In this way, we ensure no one picks up the same block until the winner releases the block (this message can be broadcast).

In addition, we notice periodical synchronization of status of all objects is necessary: Physics is simulated on individual machines, so a slight difference (due to network delay) in the time an action is done can lead to very different behavior. Again, any node may decide to ask others to sync with his world, so Paxos on a "sync_leader" key is used here.

5 Acknowledgements

We specially thank Yifei Yin for her the artwork (all models, textures, materials, etc., except the player figure) included in the game package. The LSP protocol and Paxos specification is attributed to 15-440 staff.

The following elements are found on the Internet:

- RTS-style unit selection code (used in bird's view) and the cute player figure: `http://hyunkell.com/blog/rts-style-unit-selection-in-unity-5/`
- Function for deciding current OS: `http://stackoverflow.com/a/5117005`
- Function for obtaining local IP address: `http://stackoverflow.com/a/27376368`

All other scripts and assets are work of the authors.

Bonus: Using Your Own 3D Models

Feeling artistic? You can easily design the game scene and use other 3D models. Add your favorite elements in the main scene in Unity Editor, and attach `Pickupable`, `UnitSelectionComponent`, and `RigidBody` to items (see existing boxes for parameters). For "mines" simply add a `ResourceGenerator` component to it and specify the generated object (a prefab). Enjoy!

The authors are planning to allow every user installing assets through "add-on packages" soon.