U0LPMPL2U : if you're trying do DRY your code, extract functions
U0LPMPL2U : `update` is for handling input from the outside world
U4872964V : <@U2SR9DL7Q> yes, Cmd.batch and no, it's not good practice, just do the work directly instead
U0LPMPL2U : for example:```
Foo -> (setFieldOnModel model, Cmd.batch [cmd1, cmd2])
Bar -> (setFieldOnModel model, Cmd.none)
```

U0LPMPL2U : the common model logic is extracted into the `setFieldOnModel` function and called from both branches of the `case`
U0LPMPL2U : the `Foo` branch uses `Cmd.batch` to trigger multiple side effects with commands
U2SR9DL7Q : hmmm... I had to use update so far because I needed random number generation. But now I have a sequence of operations and model changes that happen after that (for some game logic).
U2SR9DL7Q : But I think I get what you're trying to say.
U0LPMPL2U : can you post an example? It might be easier to give advice on a concrete scenario
:slightly_smiling_face:
U0LPMPL2U : when you're generating several random numbers at once, it's often better to combine the generators into into a single one that returns a tuple or record and only use a single command
U2SR9DL7Q : Sure...```
SetGame newSet firstplayer -&gt;
        let
            players =
                createPlayers newSet
        in
        ( { model | dominoes = newSet, game = Just &lt;| Game [] players firstplayer }, Cmd.none )

    ShuffleList -&gt;
        ( model, Random.generate SetFirstPlayer &lt;| shuffle model.dominoes )

    SetFirstPlayer newSet -&gt;
        ( model, Random.generate (SetGame newSet) (<http://Random.int|Random.int> 1 4) )
```
To explain, when the user clicks `Start Game`, `Shuffle List` is called and shuffles a set of domnioes. That's passed to `SetFirstPlayer` which also needs randomness to decide who goes first. That was passed to setGame to make the appropriate model updates.

U2SR9DL7Q : `SetGame` should be at the same level of indentation as `ShuffleList`.
U0LPMPL2U : Could all the randomness be done in one step? Shuffling dominoes and selecting the first player?
U2SR9DL7Q : I now realize I could've just used one Msg for all the randomness.
U23SA861Y : map is your friend
U2SR9DL7Q : But now that the game is set, it automatically starts doing things... The first player must now select a domino from their hand and place it on the board, and pass the turn to the next player, etc. This is all tracked using the model.
U2SR9DL7Q : So doesn't that mean update functions? This is what I meant by chaining updates.
U0LPMPL2U : All the setup can be a single step
U23SA861Y : but each one of those represents a state where the game must pause waiting for user input. You don't need that pause setting up the game state
U2SR9DL7Q : There's no pause unless it's the human player. Sorry, wasn't clear. For now, 3 robo players and 1 human player.
U2SR9DL7Q : But you're right. Until the human players turn, I guess I don't need another update.
U23SA861Y : you do if you want the visual representation to change
U2SR9DL7Q : I do. Each player, human or robot, selects a domino to go on the board, which has to be displayed on the screen.
U0LPMPL2U : I'd probably have a `playAITurn` generator that does all the random choices for a single AI turn, triggers the update and sets the AI's choices on the model. This triggers a redraw of view.
U0LPMPL2U : depending on who the next player is, either wait for the user to do something or `playAITurn` again
U0LPMPL2U : possibly put a sleep in there if you don't want the AIs to play too fast
U2SR9DL7Q : luckily, AI choice implmentation isnt random right now. AI plays a very low level, but methodical way.
U2SR9DL7Q : yeah, I will need the sleep I suspect
U23SA861Y : could be random too

U23SA861Y : since you can't just issue a command it's actually useful to you to require a sleep or such
U2SR9DL7Q : the random way is to have AI select a domino at random from its hand, and then check if it can be legally played. The other way is to iterate through the hand, and select the first domino that can be legally played.
U2SR9DL7Q : since the initial hand is assigned at random, I'm not sure that it's any more random doing it the first way than the second.
U23SA861Y : either way, to dispatch a message back you yourself you need to inject a sideeffect, such as sleep or random generation
U2SR9DL7Q : Yup. I guess I'm just trying to wrap my head around the structure of the next sequence of events.
U2SR9DL7Q : and where are the logical breakpoints that I need to send a message.
U2SR9DL7Q : So far it seems every time the view needs to change is a good start.
U23SA861Y : well, update should return any time the view needs to change
U2SR9DL7Q : I've made this game in python, but I was just using the command line there, so I really wanted to try making a proper UI in elm.
U2SR9DL7Q : But translating it from OOP to FP has been... a fun challenge.
U23SA861Y : well, it's a fundamentally different paradigm
U4WH8STNX : I find it's not so different when you have really SOLID code in OOP
U4WH8STNX : in my .NET code I tend to have role interfaces with one member and static classes with static methods, maybe that's just the Elm influence :smile:
U2SR9DL7Q : <@U4WH8STNX> Yeah... to an extent, some parts are just separating my classes, with attributes being the model, and methods become functions that help those attributes.
U2SR9DL7Q : but randomness becomes something that I now need to handle explicitly at the edge of my code
U4WH8STNX : it's true that Elm makes you think of the side effects because you can't accidentally introduce them - heavens :unicorn_face:
U2SR9DL7Q : <@U0LPMPL2U> <@U23SA861Y> <@U4WH8STNX> Thank you very much folks
U68N7EGVB : I have a question about importing types. I have two files: `App.elm` and `Types.elm`.
`Types.elm`:

```
module Types exposing (Model, Msg)
...

type alias Model = { mdl: Material.Model }

type Msg = Mdl (Material.Msg Msg)
```

`App.elm`:

```
...
update : Msg -&gt; Model -&gt; ( Model, Cmd Msg )
...
case msg of
  Mdl msg_ -&gt;
...
```

How to import `Msg` from `Types.elm` so that `Mdl` is visible in that pattern matching?

U57KYFW67 : `import ModuleName exposing (Msg(..))` I think
U68N7EGVB : I tried `import  Types  exposing  (Model, Msg(..))`, but it still says that `cannot find pattern Mdl`.
U57KYFW67 : ah. hmm
U2SR9DL7Q : I also use mdl, and also have a types module, with a similar set up. But I just use import Types exposing (..) and it all works <@U68N7EGVB>
U0LPMPL2U : You want to `module Types exposing (Model, Msg(..))`
U0LPMPL2U : otherwise only the `Msg` type gets exposed but not the constructor `Mdl`
U68N7EGVB : Hm. That's what I'm trying to do, but it seems like there are always those three errors: 2x `Cannot find variable Mdl` and once `Cannot find pattern Mdl`.
Importing `Model` works perfectly fine so the module is seen.

U68N7EGVB : So it looks like it's just importing `Msg` and not `Mdl`, just like you are saying.
U57KYFW67 : ah yeah