Lua Effect Documentation

Goal

To standardize naming of all lua effects to make it clear what kind of effect it is.

At a high level
- all lua functions will start with a lowercase
- card definitions should only call other lua functions, which will delegate to c# as required

Other goals:
- as few top level functions as possible
- try to use methods to free the user from having to remember how to deal with similar related types, e.g. intexpression, intcardexpression etc

Overview

The rule engine is built using a few concepts that are combined together to form the full effect system that is used to implement all the cards and effects in the game

Selectors are used to select cards, e.g. cards in the market row, cards in hand, champions with 3 or more defense etc.

Effects are used to change the game state, e.g. sacrifice, add combat etc.

We combine selectors and effects to implement most things in the game, e.g. sacrifice a card in hand.

Abilities are used to hold effects and their trigger times (i.e. when the effect happens). For example:

Gold card ability:
- Trigger: when played
- Effect: Gain 1 gold

Effects can be combined by sequenced them into longer chains of effects. For example,

Command card ability:

- Trigger: when played
- Effect: Sequence of (gain 2 gold, gain 3 combat, gain 4 health, draw a card)

Some abilities can automatically be triggered when a condition happens, e.g

Arkus's Ally Ability:

- Trigger: automatic
- Trigger condition: When Imperial ally ability is met
- Effect: Gain 6 health

Players

Predefined constants to select players:

currentPid - current player
nextPid - next player to take a turn
oppPid - opponent of current player
nextAllyPid - next ally player
nextOppPid - next opponent

Locations

To specify a location:

loc(player, ploc), where ploc is (ends with Ploc or player loc):

inPlayPloc
discardPloc
handPloc
castPloc
deckPloc
buffsPloc
skillsPloc
revealPloc - common reveal location for any cards that get revealed to both players

myRevealPloc - used when you need to reveal cards only to player

also, predefined locs without a need to specify player (ends with -Loc):

centerRowLoc
fireGemsLoc
tradeDeckLoc
revealLoc

currentInPlayLoc
currentCastLoc
currentDeckLoc
currentHandLoc
currentDiscardLoc
currentBuffsLoc
currentSkillsLoc
currentRevealLoc

Examples:

```
-- these two are the same
currentInPlayLoc
loc(currentPid, inPlayPloc)
```

Factions

```
wildFaction
guildFaction
necrosFaction
imperialFaction
```

BoolCardExpressions (TODO: check if all expressions are present)

They all start with isCard-

isCardExpended()
isCardStunned()
isCardStunnable()
isCardChampion()
isCardAction()
isCardFaction(Faction.Wild)
isCardType("Knife") -> checks type and subtype
isCardName("thief_throwing_knife") → checks card id
isCardAtLoc(ploc) → true if target is in the passed CardLocEnum

operators
.And(isCardXX()), .Or(isCardXX()), .invert()
Note that these are uppercase, as and/or are keywords in lua

constBoolExpression(value) returns a BoolExpression that always returns the value it was initially passed.

Usage
Bool card expressions are mostly used to filter out data from selectors (see below)

Examples:

```
-- selects cards from opponent's in play, which are not expend
ed and can be stunned


selectLoc(loc(oppPid, inPlayPloc)).where(isCardExpended().inve
rt().And(isCardStunnable()))
```

Selectors TODO: check if there are more predefined selectors

Selectors are used to start a chain of selecting cards. All selectors start with the selectX:

selectLoc(loc): select cards in a location
selectSource(): select source card
selectTargets(): selects target from targeted effect or cardeffect ability trigger.
selectOppStunnable(): select all opponent’s champions that can be stunned

Filtering

selector.where(BoolCardExpression) to filter by a property on each card

Example:

```
-- selects cards from opponent's in play, which are not expend
ed and can be stunned


selectLoc(loc(oppPid, inPlayPloc)).where(isCardExpended().inve
rt().And(isCardStunnable()))
```

Chaining

selector1.union(selector2) to return the union of both selectors.

```
-- returns cards from the market and fire gems
selectLoc(centerRowLoc).union(selectLoc(fireGemsLoc)).where(is
CardAffordable().And(isCardAcquirable()))
```

selector.exclude(selectSource()) to exclude the source card

Ordering

selector.order(intcardexpression) to order by the provided value
.orderDesc(intcardexpression)

.reverse() to reverse the order

selector.take(n) to take the first X cards

.take(intExpression)

Conversions

To Int: .count()

or
.sum(intcardexpression) , e.g. .sum(getCardCost())

You can use .take(1).sum(getCardCost()) to get the cost of a single card, e.g. for comparison

Predefined selectors

We predefine commonly used selectors in lua

selectCurrentChampions()
selectNextChampions()

Examples

to select all stunned champions

selectCurrentChampions().where(isCardStunned())

to count all cards in opponent's hand

selectLoc(loc(nextPid, handPloc)).count()

action cards in trade row

selectLoc(centerRowLoc).where(isCardAction())

IntExpression TODO: check if we have all int expressions specified here

Most int expressions start with get-

From selector: selector.count()

getCounter(pid, counter)
getPlayerHealth(pid)
getPlayerCombat(pid)
getPlayerGold(pid)
getPlayerDamageReceivedThisTurn(pid)
getTurnsPlayed(pid)

constant int:
const(int)

Arithmetic
.add(intExpression), .multiply(intExpression), negate()
use negate with add to subtract

Conversions:
.gte(intExpression or int), .lte(intExpression or int), .eq(intExpression or int)

Used when you need to check for something - returns BoolExpression

Conditional:
ifInt(boolExpression, intExpression1, intExpression2) - if bool true - returns expression1, otherwise expression 2

Operations:

minInt(intExp, intExp) - returns min of two values

StringExpression

There are several places where we can show dynamic strings. To create a dynamic string, just use the format function.

format(“{0}”, { int, intexpressoin etc})

IntCardExpression

These start with getCard-

getCardCost()
getCardHealth()

BoolExpression

hasClass(playerExpression, heroClass)
hasPlayerSlot(playerExpression, slot)
constBoolExpression(value) - constant value

Negate

.invert() to negate, e.g. isExpended().invert() returns false if the card is expended (not is a reserved word in lua)

Combining

.And(boolExp), .Or(boolExp)

Extended examples TODO: add more examples for selectors

Select all cards in trade row where their cost is greater than the players’ gold

selectTradeRow().where(getCardCost().gt(getPlayerGold(currentPid)))

Note that getCardCost() is a IntCardExpr, the gt() accepts an IntExpr, which should get upgraded to an IntCardExpr, and the entire expression becomes a boolcardexpr

Effects

Simple Effect TODO: check if all available effects are present here

All simple effects end with the word -Effect. These require no input from user.

- endGameEffect()
- showMessageEffect(message)

We can chain simple effects together with

effect1.seq(effect2).seq(effect3)

We can repeat simple effect execution with

effect1.doRepeat(intExpression)

Predefined simple effects:

Game state effects

- gainCombatEffect(int(expression))
- gainGoldEffect(int(expression))
- gainHealthEffect(int(expression))
- drawCardsEffect(int(expression))
- drawToLocationEffect(intExpression, locExpression) - draws count cards to location
- hitOpponentEffect(int(expression)) - deals damage to opponent
- hitSelfEffect(int(expression)) - deals damage to self
- oppDiscardEffect(int(expression)) - makes opponent discard cards when their turn starts
- createCardEffect(cardId, location) - creates a card at location
- endGameEffect(winnerId, endReason) - ends game with a winner and given reason
- setMarketLengthEffect(int) - sets length of market row to the value

- incrementCounterEffect(string counterId, int(Expression)) - increments given counter by the incoming integer value
- shuffleEffect(locExr) - shuffles all cards contained in the passed location

Conditionals
conditions are bool expressions

- ifEffect(condition, simpleEffect) - if condition is true, executes the effect
- ifElseEffect(condition, yesSimpleEffect, noSimpleEffect) - if condition is true executes yes effect, otherwise no effect

Specials

- noUndoEffect() - prevents user from undoing their actions from this point
- nullEffect() - does nothing - can be used to do nothing in certain cases

Fatigue counter

fatigueCount(startingTurn, delta, name) - fatigue counter

```
local oneSwitch = equalSwitchEffect(
    s.GetCounter(name),
-- default effect if counter value doesn't match 0 or 1 = 2 in power of counter - 1
-- in case of 2 - which is first default value hit after starting turn - 2^(2-1) = 2
    getFatigueEffect(s.Power(s.Const(2), s.Minus(s.GetCounter(name), s.Const(1)))),
    e.ValueItem(0, e.NullEffect()),
    e.ValueItem(1, getFatigueEffect(s.Const(1)))
)
```

Random selection

- randomEffect({valueItem1, valueItem2, ...}) - value item is an integer/simple effect pair. Integers represent weight of the given item. Say we have 5, 2, 1 items in the list. item with 5 has the most chances to be selected.
- valueItem(int, effect) - creates value item for RandomEffect, where determines probability of an effect to be randomly selected.

```
return randomEffect({
    valueItem(5, effect1),
    valueItem(5, effect2),
    valueItem(5, effect3)
})
```

- randomChoice(choicesArray1, choicesArray2)

```
return randomChoiceEffect({
    choices = {
       {
            effect = effect,
            layout = createLayout()
       },
       {
            effect = effect,
            layout = createLayout()
       }
     },{
    choices_2 = {
       {
            effect = e.CreateFromString("fire_gem", currentHan
dLoc),
            layout = createLayout()
       },
       {
```

```
            effect = e.CreateFromString("fire_gem", currentHan
dLoc),
            layout = createLayout()
        }
    }
})
```

Control flow

- equalSwitchEffect(intExpression, defaultSimpleEffect, valueItems[]) - value item is an integer/simple effect pair. Integers represent value of incoming integer to be compared to. Effect from value item with matching integer will be executed. If none of the items match, default effect is executed.

Animations / text

- animateShuffleEffect(player) - plays deck shuffle animation
- showOpponentTextEffect(text) - shows text to the opponent
- hideOpponentTextEffect(text) - hides previously shown opponent text
- showCardEffect(layout) - you may build a layout to be show to player to several seconds or until clicked. createLayout("avatars/troll", "Injured Troll I", "{3 combat}", "The troll gets angry")
- showTextEffect(text) - shows text on the screen
- waitForClickEffect(playerText, oppText)- shows text to corresponding player and waits for click
- customAnimationEffect({id, player}) - plays a custom animation with “id” for “player”
- cardAbilityAnimationEffect({ id, layout }) - plays card animation with “id” style and “layout” card face

Card Effects TODO: check if all effects are listed

Card effects require a list of cards to be supplied, normally via a selector. They end with -Target

Game state card effects

- sacrificeTarget() - sacrifices targets
- acquireForFreeTarget(loc) - acquires targets for free to location
- acquireTarget(int discount, loc) - acquires targets to location with a discount
- addSlotToTarget(slot) - adds a slot to targets
- damageTarget(intExpression) - deals damage to targets
- discardTarget() - discards targets
- expendTarget() - expends targets
- grantHealthToTarget(intExpression) - grants +X defense to target champions
- moveTarget(locExr) - moves targets to location
- moveToTopDeckTarget() - moves targets to top of current player's deck
- nullTarget() - does nothing
- playTarget() - plays targets
- prepareTarget() - prepares targets
- stunTarget() - stuns target
- transformTarget(cardStringId) - transforms target to a card with the given id
- modifyCardDefTarget(healthDelta, ability) - gives an existing card a modified health value and/or a new ability

Random card effects

- probabilityTarget({ chance, onSuccessTarget, onFailureTarget}) - executes onSuccess target effect with chance probability, otherwise executes onFalureTarget effect.
- randomTarget(cardEffect, intExpression) - passes X random targets to underlying card effect

Animation card effects

- showTextTarget(text) - show the provided text above the target card

Control flow card effects

- ifElseTarget(boolCardExpression, yesCardEffect, noCardEffect) - same as ifElseEffect

Card Effect can be converted into Simple Effect by applying a selector.

Examples:

Sacrifice all cards in the current player's hand:

sacrificeTarget().apply(selectLoc(loc(currentPid, handPloc)))

We can also convert a simple effect into a card effect by ignoring the target:

ignoreTarget(simpleeffect)

Targeted Effects

Targeted effects can be pushed into the effect queue with pushTargetedEffect(). They will prompt the player to select a target, and then execute the provided card effect on the targets.

Note that pushTargetedEffect() will return a Simple Effect.

If you call pushTargetedEffect in the middle of a sequence of effects, the pushed effect will not happen until everything else in the sequence has already happened. This means if you want another effect to not happen until after the targeted effect has executed, that other effect must be included in the "targetEffect" field of the table passed to pushTargetedEffect(), e.g. targetEffect=sacrificeTarget().seq(drawCardsEffect(1)).

Also, note that you can use selectTargets() to return the targets while processing a targeted/card effect.

Example to prompt the user to sacrifice a card in hand:

```
pushTargetedEffect({
  desc="Sacrifice a card in hand",
  min=0,
  max=1,
  validTargets=selectLoc(loc(currentPid, handPloc)),
```

```
  targetEffect=sacrificeTarget(),
  tags = { "cheapest" }
})
```

Choice Effects

The user will make a choice, then a SimpleEffect will be executed for the chosen item.

```
pushChoiceEffect({
  choices={
    {
      id="choicelayoutid",
      effect=gainCombatEffect(5)
    },
    {
      effect = healPlayerEffect(currentPid, v),
      layout = layoutCard({
          title = "Heal Myself",
          art = "icons/cleric_lesser_resurrect",
          text = format("{{{0} health}}", { v }),
          flavor = format("Health: {0}", { getPlayerHealth(cur
rentPid) })
      })
    }
  }
})
```

If you are using the field "text" in layoutCard ({ }) and
want to display the gold/combat/health icon in the description
use {X gold} ; {X combat} ; {X health} where X - count of points

Example:

```
layout = layoutCard({
    title = "Hunter's Cloak",
    art = "icons/ranger_hunters_cloak",
    text = ("{2 health}")
}),
```

Note that choice effects can be dynamically generated using the layoutCard() function, which returns a dynamic layout for the art. All the fields of layoutCard accept StringExpression.

Target Player effects (not implemented yet)

In coop, we might need to target players.

```
pushTargetPlayerEffect({
  mode = targetPlayers,
  effect = playerGainHealthEffect(targetPid, 5)
})
```

targetPid will be defined within the context of a target player effect.

mode can only be targetPlayers for now.

Creating abilities

Abilities link effects with their trigger times.

createAbility({id, trigger, effect, cost, activations, check})

id: unique id of the ability (within a card)

trigger: one of

```
startOfTurnTrigger
endOfTurnTrigger
endOfOppTurnTrigger
oppStartOfTurnTrigger
autoTrigger - automatic trigger, like death cultist's expend a
bility has auto trigger
onExpendTrigger
onSacrificeTrigger
uiTrigger - user will need to manually trigger it
onAcquireTrigger - triggers every time you acquire any card, n
ot just the card with the onAcquire ability
onPlayTrigger - triggers every time you play any card, not jus
t the card with the onPlay ability
startOfGameTrigger
endOfGameTrigger
deckShuffledTrigger = ActivationTrigger.DeckShuffled
```

effect: an effect to run when triggered

cost: requirements to use the effect, set if required; to give an ability multiple costs, such as expending and also paying gold, use combineCosts(t), passing it a table that contains each of the desired costs, e.g. combineCosts({ expendCost, goldCost(2) })

- expendCost
- sacrificeSelfCost
- goldCost(value)
- sacrificeSelfCost
- costAnd({ expendCost, goldCost(value) })

activations: set if required

- singleActivation: normally used with auto abilities to indicate that the ability can only be activated once per turn. Default.

- multipleActivations: used with expend abilities to indicate that it can be activated multiple times per turn.

check: bool expression to decide if the ability can be triggered

allyFactions: used for Ally Abilities, which can only activate if a card of the specified faction(s) is also in play; required for createUiAllyAbility() and createAutoAllyAbility()

Creating CardEffectAbilities

Standard Abilities can have the onAcquire trigger, which makes them activate whenever *any* card is acquired (regardless of destination), which can make it difficult to have the ability affect the acquired card, as the onAcquire ability does not have a reference to it. If you want to create an ability that will always be able to affect the specific card that was acquired, you need to give the Card Def a CardEffectAbility. The createDef() function accepts a cardEffectAbilities table in its table parameter, though the abilities table is still required, even if a card has a CardEffectAbility but no Ability:

```
card = createDef({
  ...
  abilities = { },
  cardEffectAbilities = {
    createCardEffectAbility({
      trigger = playedCardTrigger,
      effect = expendTarget().apply(selectTargets().where(isCa
rdChampion()))
    })
  },
  ...
})
```

There are several possible trigger timings for a CardEffectAbility:

- acquiredCardTrigger - triggers before card is acquired
- locationChangedCardTrigger - triggers when a card is added to the market, created, or moved

- postAcquiredCardTrigger - triggers after card is acquired
- postSelfAcquiredCardTrigger - like postAcquired, but only triggers for the card itself
- playedCardTrigger - triggers after a card is played from hand. The trigger will happen after after played triggers, but before auto triggers happen. This way, if the played card trigger expends a champion, auto triggers will not fire.

CardEffectAbilities must use CardEffects, which require specific card targets. The CardEffectAbility will pass the acquired card as the target for the CardEffect assigned to it if the trigger is onAcquire or postAcquire; it will pass the card whose location changed if the trigger is onLocationChanged. You can also use selectTargets() to obtain the targets

Additionally, for a card that contains a CardEffectAbility with the onLocationChanged trigger, when that card is first created, that ability will trigger, targeting every possible card.

You can pass a Simple Effect to the CardEffectAbility, it will be auto converted to a CardEffect.

Creating Card Defs

Finally, we put the abilities into a Card Def to allow us to actually create a card.

```
createDef({id, name, cardTypeLabel, playLocation, abilities, types, ….})
```

id: a unique id for the card

name: Title of card

abilities: array of abilities that the card will have

cardEffectAbilities: array of card effect abilities

acquireCost: cost to acquire

types: array of strings

health: health/defense for champions

healthType: for champions, its either defenseHealthType (default) or healthHealthType

buffDetails: for global buffs, this creates the display when its clicked for details:

```
createBuffDetails({
            art = "wizard_spell_components",
            name = "Soak",
            text = "+1 cost"
        }),
```

layout: if we want to dynamically generate the card layout, example code:

```
createLayout({
            name = "Little Fire Sacer",
            art = "icons/fighter_knock_back",
            text = "Expend: Sacrifice a card in your hand or d
iscard pile",
            flavor = "Water cleanses all"
        })
```

See layout text chapter below for more formatting info.

Alternatively, we can load a card texture as layout, e.g.
`loadLayoutTexture("Textures/death_touch")`

cardTypeLabel: type of card, used only for display. Normally auto filled in.

playLocation: where the card is played, one of the player loc, e.g. castPloc for action cards. . Normally auto filled in.

Helper methods for creating card defs. These generally just fill in the required type, playLocation and cardTypeLabel.

createActionDef()

```
function confused_apparition_carddef()
    return createActionDef({
        id="confused_apparition",
        name="Confused Apparition",
        types={noStealType},
        acquireCost=0,
        abilities = {
            createAbility({
                id="confused_apparition_auto",
                trigger= autoTrigger,
                effect = ifEffect(selectLoc(currentInPlayLoc).
where(isCardName("weak_skeleton")).count().lte(0), healPlayerE
ffect(oppPid, 1))
            })
        },
        layout = createLayout({
            name = "Confused Apparition",
            art = "art/T_Confused_Apparition",
            frame = "frames/Coop_Campaign_CardFrame",
            text = "Opponent gains 1 <sprite name=\"health\">
unless you have a Weak Skeleton in play."
        })
    })
end
```

createChampionDef()

```
function orc_guardian_carddef()
    return createChampionDef({
        id="orc_guardian",
```

```
        name="Orc Guardian",
        types={orcType, noStealType},
        acquireCost=0,
        health = 3,
        isGuard = true,
        abilities = {
            createAbility({
                id="feisty_orcling_auto",
                trigger=autoTrigger,
                effect = e.NullEffect()
            })
        },
        layout = createLayout({
            name = "Orc Guardian",
            art = "art/T_Orc_Guardian",
            frame = "frames/Coop_Campaign_CardFrame",
            text = "<i>He's quite defensive.</i>",
            health = 3,
            isGuard = true
        })
    })
end
```

createBuffDef()

createSkillDef()

```
function piracy_carddef()
    return createSkillDef({
        id="piracy",
        name="Piracy",
        abilities = {
```

```
            createAbility({
                id="piracy_auto",
                trigger=autoTrigger,
                effect = --showTextTarget("Piracy!").apply(sel
ectSource())
                        showCardEffect(layoutCard({
                            title = "Piracy",
                            art = "art/T_Piracy",
                            frame = "frames/Coop_Campaign_Card
Frame",
                            text = "Acquire the cheapest card
in the market row for free"
                        }))
                        .seq(acquireForFreeTarget().apply(sele
ctLoc(centerRowLoc).where(isCardAcquirable()).order(getCardCos
t()).take(1)))
                        .seq(ifEffect(selectLoc(currentDiscard
Loc).reverse().take(1).sum(getCardCost()).gte(6), showTextEffe
ct("Mighty fine plunder, that one.")))
            })
        },
        layout = createLayout({
            name = "Piracy",
            art = "art/T_Piracy",
            frame = "frames/Coop_Campaign_CardFrame",
            text = "Acquire the cheapest card in the market ro
w for free"
        })
    })
end
```

createMagicArmorDef()

```
function cleric_shining_breastplate2_carddef()
    local cardLayout = createLayout({
        name = "Shining Breastplate 2",
        art = "icons/cleric_shining_breastplate",
        frame = "frames/Cleric_CardFrame",
        text = "Champion get +1 defense till the end of turn"
    })

    return createMagicArmorDef({
        id = "cleric_shining_breastplate2",
        name = "Shining Breastplate 2",
        types = {clericType, magicArmorType, treasureType, che
stType},
        layout = cardLayout,
        layoutPath = "icons/cleric_shining_breastplate",
        abilities = {
            createAbility( {
                id = "cleric_shining_breastplate2",
                trigger = uiTrigger,
                activations = singleActivation,
                layout = cardLayout,
                effect = pushTargetedEffect(
                    {
                        desc = "Choose a champion to get +1 de
fense",
                        validTargets =  s.CurrentPlayer(CardLo
cEnum.InPlay),
                        min = 1,
                        max = 1,
                        targetEffect = grantHealthTarget(1, {
SlotExpireEnum.LeavesPlay }, nullEffect(), "shield"),
```

```
                        tags = {toughestTag}

                    }
                ),
                cost = AbilityCosts.Expend,
                check = minHealthCurrent(40).And(selectLoc(cur
rentInPlayLoc).where(isCardChampion()).count().gte(1))
            })
        }
    })
end
```

createHeroAbilityDef()

Layout Text formatting

Formatting text in card layout should follow TextMeshPro unity component guidelines.

`voffset` gives the baseline a vertical offset. You can use pixels or font units and it's always relative to the original baseline. The closing tag resets back to the original baseline.

The line height is adjusted to accommodate the displaced text. If you don't want that, you can manually adjust the line height.

The `pos` tag gives you direct control over the horizontal caret position. You can put it anywhere on the same line, regardless where it started. You can use either pixels, font units, or percentages.
This tags is best used with left alignment.

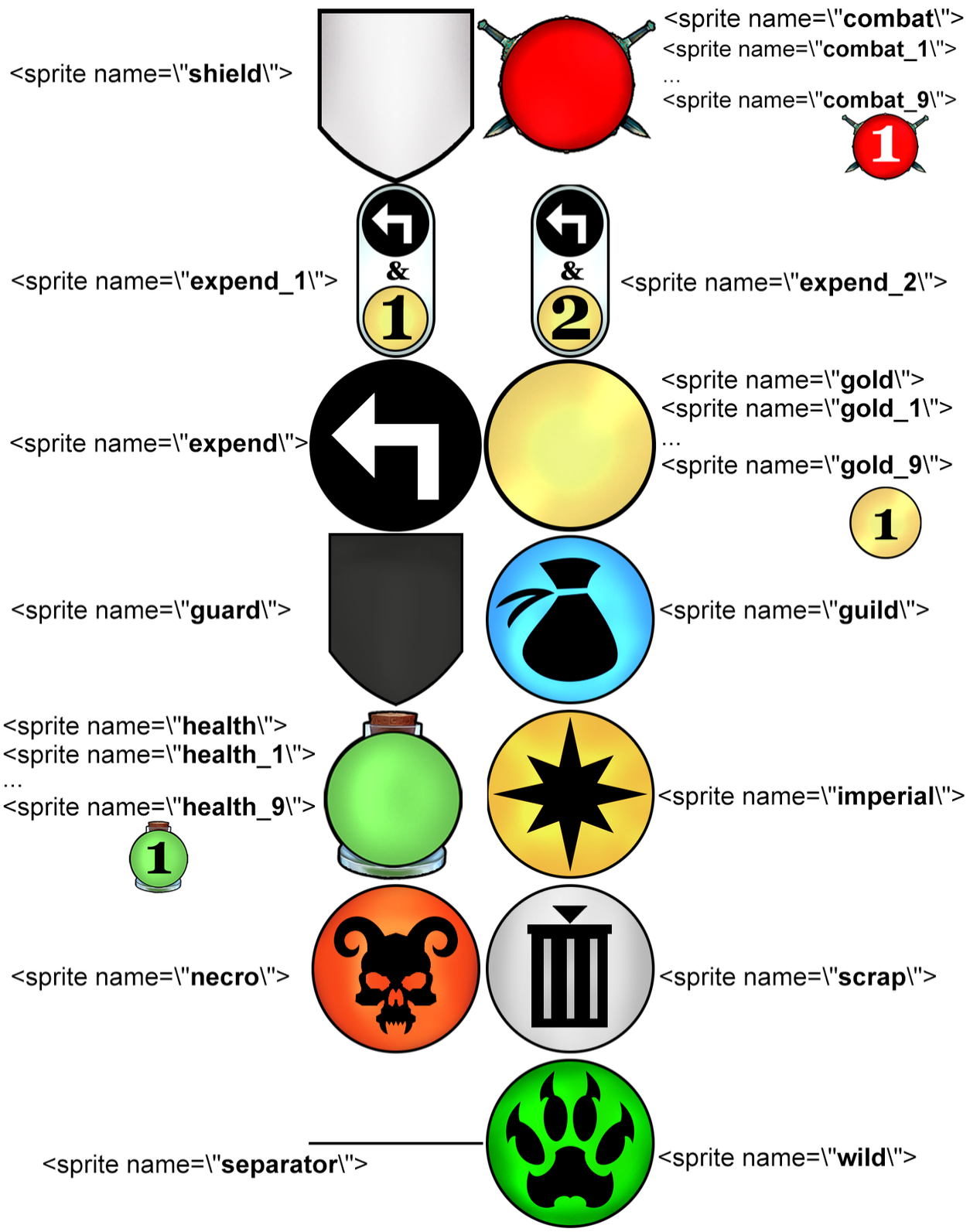
The `line-height` tag gives you manual control over the line height. Use it to pull lines closer together or push them further apart. As the line-height controls how far down the next line start, this tag does not change the current line.

You can adjust the font `size` of your text at any time. You can specify the new size in either pixels, font units, or as a percentage. Pixel adjustments can be either

absolute or relative, like `+1` and `-1`. All relative sizes are based on the original font size, so they're not cumulative.

The `space` tag inserts a horizontal offset, as if you inserted multiple spaces. You can use pixels or font units.
This tag interacts with word wrapping by sticking to the words it touches. If you want them to word-wrap separarately, put space characters around this tag.

All other tags that can also be used are detailed in the documentation with examples:
http://digitalnativestudios.com/textmeshpro/docs/rich-text/#size



Sprites:

<sprite name="shield">

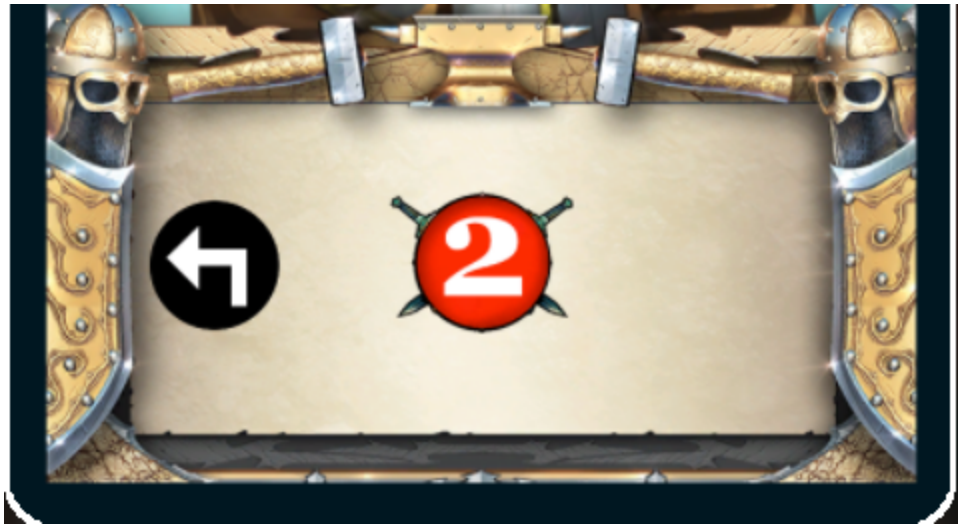
<sprite name="combat"> <sprite name="combat_1">
<sprite name="expend_1">
<sprite name="expend_2">
<sprite name="expend">
<sprite name="gold"><sprite name="gold_1">
<sprite name="guard">
<sprite name="guild">
<sprite name="health"><sprite name="health_1">
<sprite name="imperial">
<sprite name="necro">
<sprite name="scrap">
<sprite name="separator">
<sprite name="wild">

Example:

```
"<size=300%><line-height=0%><voffset=-0.6em> <pos=-75%><sprite name=\"requiresHealth_40\"></size><line-height=100%> \n <voffset=1.5em>Target<space=2em> \n <space=0.1em> Champion \n  gets +1 <sprite name=\"shield\"> permanently."
```

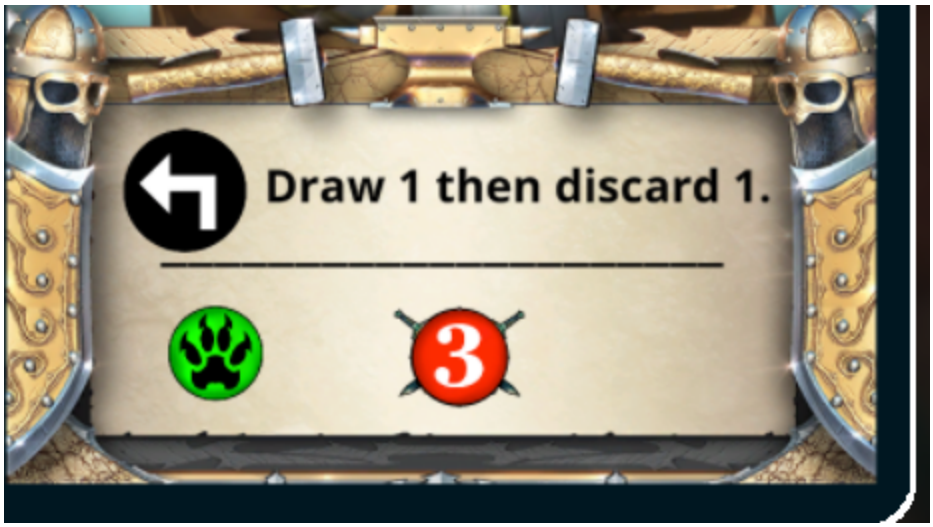

```
"Gain 2 <sprite name=\"health\"> and 1 <sprite name=\"combat\"> for each time your deck has been shuffled"
```

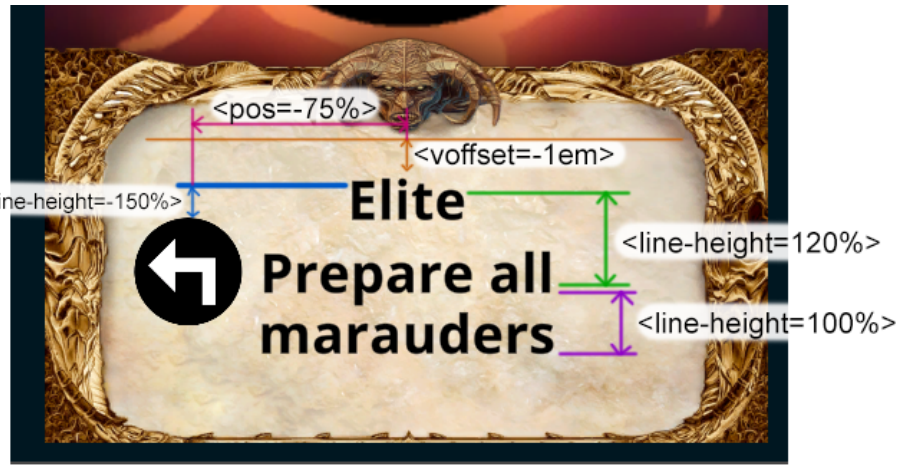

```
"<size=200%><space=-2em><voffset=-4><sprite name=\"expend\"></
voffset><space=1em><sprite name=\"combat_2\"></size>",
```



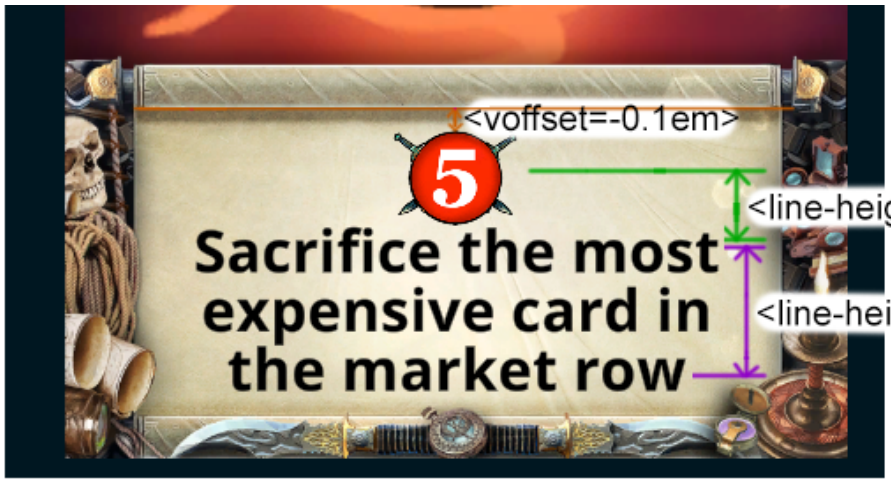

```
"Elite<br><voffset=1em><space=-3.7em><voffset=0.2em><size=200%
><sprite name=\"expend\"></size></voffset><pos=30%> <voffset=
0.5em><line-height=40><space=-3.7em>Stun a<br>champion</voffse
t></voffset>"
```


```
"<line-height=175><size=190%><voffset=-140><pos=5><sprite name
=\"expend\"></pos></voffset></size><size=70%><voffset=-125><po
s=70>Draw 1 then discard 1.</pos></voffset></size><br><voffset
=25><pos=0>_____</pos></voffset><br><voffset=1
40><pos=-95><size=143%><sprite name=\"wild\">      <sprite nam
e=\"combat_3\"></size></pos></voffset>"
```

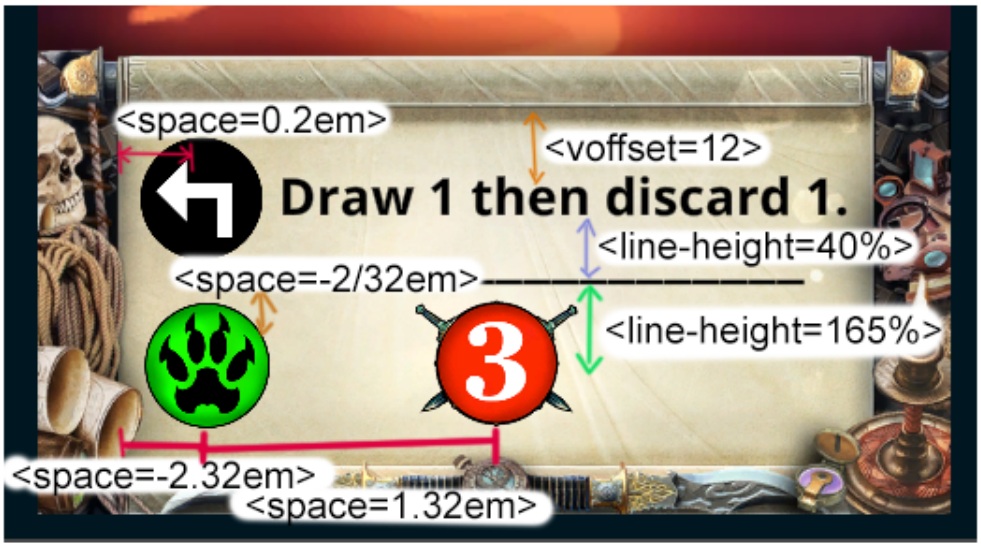





```
<voffset=-1em><pos=-75%>
    <line-height=-150%>
        <size=200%><sprite name=\"expend\"></size><br>
    </line-height>
    <line-height=120%>
        Elite<br>
    </line-height>
    <line-height=100%>
        Prepare all<br>
        marauders
    </line-height>
</voffset>
```




```
<voffset=-0.1em>
  <line-height=100%>
    <size=150%><sprite name=\"combat_5\"></size><br>
  </line-height>
  <line-height=90%>
    Sacrifice the most expensive card in the market row
  </line-height>
</voffset>
```




```
<size=180%>
   <sprite name=\"expend\">
</size>
<size=75%>
   <voffset=12>
      <space=0.2em>Draw 1 then discard 1.
   </voffset>
</size>
<line-height=40%><br>_____</line-height>
<line-height=165%><br>
   <size=180%><space=-2.32em>
      <voffset=-4>
         <sprite name=\"wild\">
      </voffset>
      <space=1.32em>
      <size=170%><sprite name=\"combat_3\">
   </size>
</line-height>"
```

Slots TODO: add slots info

There are two types of slots in the engine:

- card slots
- player slots

Slots are customizable objects that can be attached to cards and players for various purposes.
All slots have a lifespan, determined by expiration field.

Slot Expiry

```
startOfOwnerTurnExpiry
endOfOwnerTurnExpiry
endOfTurnExpiry
leavesPlayExpiry
linkedExpiry
neverExpiry
```

Card slots

Card slots are used to change card properties in various ways.

Available slot types:

Factions Slot
When this slot is attached to a card, it's considered to have factions specified and this effect falls of when expiration event comes.

```
createFactionsSlot({ necrosFaction, guildFaction }, { leavesPl
ayExpiry, endOfTurnExpiry })
```

Ability Slot
When this slot attached to a card, it also has the ability in it.

```
local ab = createAbility({
    id = "expendio",
    effect = prepareTarget().apply(selectSource()),
```

```
    cost = noCost,
    check = selectSource().where(isCardExpended()).count().eq
(1),
    trigger = autoTrigger,
    activations = singleActivation,
    tags = {  }
})

createAbilitySlot(ab, { endOfTurnExpiry })
```

Cost change Slot
Modifies cost of a card it attached to.

```
createCostChangeSlot(discount, expiresArray)
```

if discount is negative, card will cost more.

NoBuy Slot
When this slot is attached to a card in the market row, a card can't be acquired even if you have enough gold.

```
createNoBuySlot(expires)
```

Health change Slot
Modifies defense of a card it attached to.

```
createCostChangeSlot(discount, expiresArray)
```

if discount is negative, card will cost more.

Player slots (coming soon)

Player slots can be attached to players and then can be checked and used to make decisions.
Player slot consists of a key and optional string value.

For example, slots are used to mark a player as one who has a champion stunned this turn, so phoenix helm could activate.

Examples: TBD

AI

There are 3 types of AI available in the app:

- createEasyAi()
- createMediumAi()
- createHardAi()

Image resources

Those can be art, frames, icons and avatars.
Virtually all arts except frames are interchangeable.
You may use avatars for choices and layouts, but not vice versa.
Path should be set as a path, see exact values below.

Art

Art folder is used to generate layouts. It determines the art on the card.

Possible values:

```
art/dark_sign
art/Gorg__Orc_Shaman
art/T_Angry_Skeleton
art/T_Barreling_Fireball
art/T_Battle_Cry
art/T_Bounty_Collection
art/T_Broadsides
art/T_Capsize
```

```
art/T_Captain_Goldtooth
art/T_Confused_Apparition
art/T_Cunning_Of_The_Wolf
art/T_Death_Touch
art/T_Devotion
art/T_Evangelize
art/T_Expansion
art/T_Explosive_Fireball
art/T_Feisty_Orcling
art/T_Fire_Bomb
art/T_Fire_Gem
art/T_Flesh_Ripper
art/T_Heist
art/T_Influence
art/T_Life_Force
art/T_Maroon
art/T_Maurader
art/T_Orc_Guardian
art/T_Orc_Riot
art/T_Pillar_Of_Fire
art/T_Piracy
art/T_Rolling_Fireball
art/T_Seek_Revenge
art/T_Set_Sail
art/T_Strength_In_Numbers
art/T_Strength_Of_The_Wolf
art/T_The_Rot
art/T_TimelyHeist
art/T_Weak_Skeleton
```

Frames

Frames are used to set a frame for a card layout

Possible values:

```
frames/Cleric_armor_frame
frames/Cleric_CardFrame
frames/Fighter_armor_frame
frames/HR_CardFrame_Action_Guild
frames/HR_CardFrame_Action_Imperial
frames/HR_CardFrame_Action_Necros
frames/HR_CardFrame_Action_Wild
frames/HR_CardFrame_Champion_Guild
frames/HR_CardFrame_Champion_Imperial
frames/HR_CardFrame_Champion_Necros
frames/HR_CardFrame_Champion_Wild
frames/HR_CardFrame_Item_Generic
frames/Necros_Action_CardFrame
frames/Ranger_armor_frame
frames/Ranger_CardFrame
frames/Thief_armor_frame
frames/Thief_CardFrame
frames/Warrior_CardFrame
frames/Wild_Champion_CardFrame
frames/Wizard_armor_frame
frames/Wizard_CardFrame
```

Icons
Icons may be used for choice layout generation or to set icons for buffs, skills and abilities

Possible values:

```
icons/battle_cry
```

```
icons/cleric_battle_resurrect
icons/cleric_bless
icons/cleric_bless_of_heart
icons/cleric_bless_of_iron
icons/cleric_bless_of_soul
icons/cleric_bless_of_steel
icons/cleric_bless_the_flock
icons/cleric_brightstar_shield
icons/cleric_divine_resurrect
icons/cleric_holy_resurrect
icons/cleric_lesser_resurrect
icons/cleric_mass_resurrect
icons/cleric_minor_resurrect
icons/cleric_phoenix_helm
icons/cleric_resurrect
icons/cleric_righteous_resurrect
icons/cleric_shining_breastplate
icons/cunning_of_the_wolf
icons/dark_sign
icons/evangelize
icons/fighter_crushing_blow
icons/fighter_crushing_blow_OLD
icons/fighter_devastating_blow
icons/fighter_devastating_blow_OLD
icons/fighter_group_tackle
icons/fighter_group_tackle_OLD
icons/fighter_helm_of_fury
icons/fighter_knock_back
icons/fighter_knock_back_OLD
icons/fighter_knock_down
```

```
icons/fighter_knock_down_OLD
icons/fighter_mighty_blow
icons/fighter_mighty_blow_OLD
icons/fighter_powerful_blow
icons/fighter_powerful_blow_OLD
icons/fighter_precision_blow
icons/fighter_precision_blow_OLD
icons/fighter_shoulder_bash
icons/fighter_shoulder_bash_OLD
icons/fighter_shoulder_crush
icons/fighter_shoulder_crush_OLD
icons/fighter_shoulder_smash
icons/fighter_shoulder_smash_OLD
icons/fighter_smashing_blow
icons/fighter_smashing_blow_OLD
icons/fighter_spiked_pauldrons
icons/fighter_sweeping_blow
icons/fighter_sweeping_blow_OLD
icons/fighter_whirling_blow
icons/fighter_whirling_blow_OLD
icons/fire_bomb
icons/fire_gem
icons/full_armor
icons/full_armor_2
icons/growing_flame
icons/life_siphon
icons/orc_raiders
icons/piracy
icons/ranger_careful_track
icons/ranger_careful_track_OLD
```

```
icons/ranger_fast_track
icons/ranger_fast_track_OLD
icons/ranger_flawless_track
icons/ranger_headshot
icons/ranger_headshot_OLD
icons/ranger_hunters_cloak
icons/ranger_instinctive_track
icons/ranger_instinctive_track_OLD
icons/ranger_longshot
icons/ranger_longshot_OLD
icons/ranger_quickshot
icons/ranger_quickshot_OLD
icons/ranger_relentless_track
icons/ranger_relentless_track_OLD
icons/ranger_snapshot
icons/ranger_snapshot_OLD
icons/ranger_steady_shot
icons/ranger_steady_shot_OLD
icons/ranger_sureshot_bracer
icons/ranger_track
icons/ranger_track_OLD
icons/ranger_triple_shot
icons/ranger_triple_shot_OLD
icons/ranger_twin_shot
icons/ranger_twin_shot_OLD
icons/ranger_well_placed_shot
icons/ranger_well_placed_shot_OLD
icons/smugglers
icons/strength_of_the_wolf
icons/thief_distracted_exchange
```

```
icons/thief_heist
icons/thief_heist_OLD
icons/thief_lift
icons/thief_lift_OLD
icons/thief_masterful_heist
icons/thief_masterful_heist_OLD
icons/thief_misdirection
icons/thief_pickpocket
icons/thief_pickpocket_OLD
icons/thief_pilfer
icons/thief_pilfer_OLD
icons/thief_practiced_heist
icons/thief_practiced_heist_OLD
icons/thief_shadow_mask
icons/thief_silent_boots
icons/thief_skillful_heist
icons/thief_skillful_heist_OLD
icons/thief_sleight_of_hand
icons/thief_smooth_heist
icons/thief_smooth_heist_OLD
icons/thief_swipe
icons/thief_swipe_OLD
icons/thief_theft
icons/thief_theft_OLD
icons/thief_timely_heist
icons/thief_timely_heist_OLD
icons/turn_to_ash
icons/T_Bounty_Collection
icons/wind_storm
icons/wizard_barreling_fireball
```

```
icons/wizard_calm_channel
icons/wizard_calm_channel_OLD
icons/wizard_channel
icons/wizard_deep_channel
icons/wizard_deep_channel_OLD
icons/wizard_explosive_fireball
icons/wizard_fireball
icons/wizard_fire_blast
icons/wizard_fire_blast_OLD
icons/wizard_flame_burst
icons/wizard_flame_burst_OLD
icons/wizard_pure_channel
icons/wizard_pure_channel_OLD
icons/wizard_rolling_fireball
icons/wizard_runic_robes
icons/wizard_scorching_fireball
icons/wizard_searing_fireball
icons/wizard_serene_channel
icons/wizard_serene_channel_OLD
icons/wizard_soul_channel
icons/wizard_soul_channel_OLD
icons/wizard_spellcaster_gloves
```

Avatars

You need to specify full path when using for layout.
But only avatar name “assassin”, when used as actual avatar.

Possible values:

```
avatars/ambushers
```

```
avatars/assassin
avatars/assassin_flipped
avatars/broelyn
avatars/broelyn__loreweaver
avatars/chanting_cultist
avatars/chest
avatars/cleric_01
avatars/cleric_02
avatars/cristov_s_recruits
avatars/cristov__the_just
avatars/fighter_01
avatars/fighter_02
avatars/inquisition
avatars/krythos
avatars/lord_callum
avatars/lys__the_unseen
avatars/man_at_arms
avatars/monsters_in_the_dark
avatars/necromancers
avatars/ogre
avatars/orcs
avatars/orc_raiders
avatars/origins_flawless_track
avatars/origins_shoulder_bash
avatars/pirate
avatars/profit
avatars/ranger_01
avatars/ranger_02
avatars/rayla__endweaver
avatars/rayla__endweaver_flipped
```

```
avatars/robbery
avatars/ruinos_zealot
avatars/skeleton
avatars/smugglers
avatars/spider
avatars/summoner
avatars/tentacles
avatars/the_wolf_tribe
avatars/thief_01
avatars/thief_02
avatars/troll
avatars/vampire_lord
avatars/wizard_01
avatars/wizard_02
avatars/wolf_shaman
```

Zoomed buffs

These are larger variants of some icons

```
zoomedbuffs/cleric_battle_resurrect
zoomedbuffs/cleric_bless
zoomedbuffs/cleric_bless_of_heart
zoomedbuffs/cleric_bless_of_iron
zoomedbuffs/cleric_bless_of_soul
zoomedbuffs/cleric_bless_of_steel
zoomedbuffs/cleric_bless_the_flock
zoomedbuffs/cleric_brightstar_shield
zoomedbuffs/cleric_divine_resurrect
zoomedbuffs/cleric_holy_resurrect
zoomedbuffs/cleric_lesser_resurrect
```

```
zoomedbuffs/cleric_mass_resurrect
zoomedbuffs/cleric_minor_resurrect
zoomedbuffs/cleric_phoenix_helm
zoomedbuffs/cleric_resurrect
zoomedbuffs/cleric_righteous_resurrect
zoomedbuffs/goblin_warlord
zoomedbuffs/ranger_horn_of_calling
zoomedbuffs/thief_distracted_exchange
zoomedbuffs/thief_misdirection
zoomedbuffs/thief_sleight_of_hand
zoomedbuffs/wizard_barreling_fireball
zoomedbuffs/wizard_explosive_fireball
zoomedbuffs/wizard_fireball
zoomedbuffs/wizard_rolling_fireball
zoomedbuffs/wizard_runic_robes
zoomedbuffs/wizard_scorching_fireball
zoomedbuffs/wizard_searing_fireball
zoomedbuffs/wizard_serpentine_staff
zoomedbuffs/wizard_spell_components
```

Card list

Card available:

```
arkus__imperial_dragon_carddef()
borg__ogre_mercenary_carddef()
bribe_carddef()
broelyn__loreweaver_carddef()
cleric_battle_resurrect_carddef()
cleric_bless_carddef()
cleric_bless_of_heart_carddef()
```

```
cleric_bless_of_iron_carddef()
cleric_bless_of_soul_carddef()
cleric_bless_of_steel_carddef()
cleric_bless_the_flock_carddef()
cleric_brightstar_shield_carddef()
cleric_divine_resurrect_carddef()
cleric_everburning_candle_carddef()
cleric_follower_a_carddef()
cleric_follower_b_carddef()
cleric_hammer_of_light_carddef()
cleric_holy_resurrect_carddef()
cleric_lesser_resurrect_carddef()
cleric_mass_resurrect_carddef()
cleric_minor_resurrect_carddef()
cleric_phoenix_helm_carddef()
cleric_prayer_beads_carddef()
cleric_redeemed_ruinos_carddef()
cleric_resurrect_carddef()
cleric_righteous_resurrect_carddef()
cleric_shining_breastplate_carddef()
cleric_spiked_mace_carddef()
cleric_talisman_of_renewal_carddef()
cleric_veteran_follower_carddef()
close_ranks_carddef()
command_carddef()
cristov__the_just_carddef()
cron__the_berserker_carddef()
cult_priest_carddef()
dagger_carddef()
darian__war_mage_carddef()
```

```
dark_energy_carddef()
dark_reward_carddef()
death_cultist_carddef()
death_threat_carddef()
death_touch_carddef()
deception_carddef()
dire_wolf_carddef()
domination_carddef()
elixir_of_concentration_carddef()
elixir_of_endurance_carddef()
elixir_of_fortune_carddef()
elixir_of_strength_carddef()
elixir_of_wisdom_carddef()
elven_curse_carddef()
elven_gift_carddef()
fighter_crushing_blow_carddef()
fighter_devastating_blow_carddef()
fighter_double_bladed_axe_carddef()
fighter_group_tackle_carddef()
fighter_hand_scythe_carddef()
fighter_helm_of_fury_carddef()
fighter_jagged_spear_carddef()
fighter_knock_back_carddef()
fighter_knock_down_carddef()
fighter_longsword_carddef()
fighter_mighty_blow_carddef()
fighter_powerful_blow_carddef()
fighter_precision_blow_carddef()
fighter_rallying_flag_carddef()
fighter_seasoned_shield_bearer_carddef()
```

```
fighter_sharpening_stone_carddef()
fighter_shield_bearer_carddef()
fighter_shoulder_bash_carddef()
fighter_shoulder_crush_carddef()
fighter_shoulder_smash_carddef()
fighter_smashing_blow_carddef()
fighter_spiked_pauldrons_carddef()
fighter_sweeping_blow_carddef()
fighter_throwing_axe_carddef()
fighter_whirling_blow_carddef()
fire_bomb_carddef()
fire_gem_carddef()
goblin_carddef()
goblin_warlord_carddef()
gold_carddef()
gold_male_black_carddef()
gold_male_dark_carddef()
gold_male_pale_carddef()
gold_male_white_carddef()
gold_female_black_carddef()
gold_female_dark_carddef()
gold_female_pale_carddef()
gold_female_white_carddef()
grak__storm_giant_carddef()
hit_job_carddef()
influence_carddef()
intimidation_carddef()
kraka__high_priest_carddef()
krythos__master_vampire_carddef()
life_drain_carddef()
```

```
lightning_gem_carddef()
lys__the_unseen_carddef()
man_at_arms_carddef()
master_weyan_carddef()
myros__guild_mage_carddef()
nature_s_bounty_carddef()
orc_grunt_carddef()
parov__the_enforcer_carddef()
potion_carddef()
potion_of_power_carddef()
profit_carddef()
rake__master_assassin_carddef()
rally_the_troops_carddef()
rampage_carddef()
ranger_black_arrow_carddef()
ranger_careful_track_carddef()
ranger_fast_track_carddef()
ranger_flashfire_arrow_carddef()
ranger_flawless_track_carddef()
ranger_headshot_carddef()
ranger_honed_black_arrow_carddef()
ranger_horn_of_calling_carddef()
ranger_hunters_cloak_carddef()
ranger_hunting_bow_carddef()
ranger_instinctive_track_carddef()
ranger_light_crossbow_carddef()
ranger_longshot_carddef()
ranger_pathfinder_compass_carddef()
ranger_quickshot_carddef()
ranger_relentless_track_carddef()
```

```
ranger_snake_pet_carddef()
ranger_snapshot_carddef()
ranger_steady_shot_carddef()
ranger_sureshot_bracer_carddef()
ranger_track_carddef()
ranger_triple_shot_carddef()
ranger_twin_shot_carddef()
ranger_unending_quiver_carddef()
ranger_well_placed_shot_carddef()
rasmus__the_smuggler_carddef()
rayla__endweaver_carddef()
reality_prism_carddef()
recruit_carddef()
ruby_carddef()
shortsword_carddef()
smash_and_grab_carddef()
spark_carddef()
spiked_mace_carddef()
street_thug_carddef()
taxation_carddef()
tentacle_carddef()
tentacle_whip_carddef()
the_rot_carddef()
thief_blackjack_carddef()
thief_distracted_exchange_carddef()
thief_enchanted_garrote_carddef()
thief_heist_carddef()
thief_jewelers_loupe_carddef()
thief_keen_throwing_knife_carddef()
thief_knife_belt_carddef()
```

```
thief_lift_carddef()
thief_masterful_heist_carddef()
thief_misdirection_carddef()
thief_pickpocket_carddef()
thief_pilfer_carddef()
thief_practiced_heist_carddef()
thief_sacrificial_dagger_carddef()
thief_shadow_mask_carddef()
thief_silent_boots_carddef()
thief_skillful_heist_carddef()
thief_sleight_of_hand_carddef()
thief_smooth_heist_carddef()
thief_swipe_carddef()
thief_theft_carddef()
thief_throwing_knife_carddef()
thief_timely_heist_carddef()
tithe_priest_carddef()
torgen_rocksplitter_carddef()
tyrannor__the_devourer_carddef()
varrick__the_necromancer_carddef()
web_jar_carddef()
wizard_alchemist_s_stone_carddef()
wizard_arcane_wand_carddef()
wizard_barreling_fireball_carddef()
wizard_blazing_staff_carddef()
wizard_calm_channel_carddef()
wizard_cat_familiar_carddef()
wizard_channel_carddef()
wizard_deep_channel_carddef()
wizard_explosive_fireball_carddef()
```

```
wizard_fire_blast_carddef()
wizard_fire_staff_carddef()
wizard_fireball_carddef()
wizard_flame_burst_carddef()
wizard_ignite_carddef()
wizard_magic_mirror_carddef()
wizard_pure_channel_carddef()
wizard_rolling_fireball_carddef()
wizard_runic_robes_carddef()
wizard_scorching_fireball_carddef()
wizard_searing_fireball_carddef()
wizard_serene_channel_carddef()
wizard_serpentine_staff_carddef()
wizard_silverskull_amulet_carddef()
wizard_soul_channel_carddef()
wizard_spell_components_carddef()
wizard_spellcaster_gloves_carddef()
wolf_form_carddef()
wolf_shaman_carddef()
wondrous_wand_carddef()
word_of_power_carddef()
```

Card Subtypes

Card subtypes are set to allow special processing, like knives or arrows.
You may check for it using isCardType(subType)

```
noStealType -- cannot be stolen by cards such as thief_heist
knifeType
bowType
```

```
arrowType
actionType
itemType
spellType
chestType
headType
championType
daggerType
abilityType
masterType
magicArmorType
minionType
dragonType
orcType
eliteMinionType
elfType
coinType
wolfType
gemType
monkType
rogueType
assassinType
goblinType
spearType
shoulderType
currencyType
axeType
bannerType
toolType
rangedWeaponType
```

```
meleeWeaponType
magicWeaponType
magicAmuletType
magicSuppliesType
swordType
handType
paladinType
scytheType

wizardType
fighterType
clericType
rangerType
thiefType

weaponType
inventoryType
instrumentType
candleType
elixirType
giantType
maceType
mageType
jewelryType
vampireType
holyRelicType
priestType
treasureType
warriorType
ogreType
```

```
humanType
hammerType
curseType
backType
snakeType
armType
felineType
staffType
wandType
trollType
necromancerType
tentacleType
clubType
footType
beltType
demonType
garroteType
attachmentType
noKillType -- mark champions with this sybtype to avoid AI kil
ling it (Redeemed Ruinos)
```