Magic Chance: Systems Checklist

Game Flow

Players will start at early levels whose difficulty is relatively low. The difficulty is determined by increasing the (in descending order of intensity):

- Most difficult probability concept included
- Additional probability concepts included
- Relevant Locked items in the trial
- Minor distractor locked items in the trial
- Complexity of stated ratios or coefficients in the trial (1/7 harder vs. 2)

Levels increase in difficulty as needed from the bottom of that list to the top.

Scoring is determined by LEVEL and NUM_ATTEMPTS since players will be given a second chance at incorrect trials. 1st attempt will gain full points, less for 2nd attempt, and none for failing the trial.

Progression is determined by NUM_ATTEMPTS, HINTS (viewing tooltips), and possibly MOMENTUM (to take the place of TIME)

Level Configurations

Levels generated from about 1010 level configurations will be grouped into 400? level buckets.

Scoring Formula

Scoring combines consideration of performance (60%), speed (30%) and past performance (10%), giving partial credit when 1 or 2 hints are used (50 or 20%) or when an arrangement is off by 1 from a correct solution (0% for now...but including it still for future consideration). Scores will sum up the scores of trials, in total ranging up to 65,000 (max score). The formula are as follows:

// Setting up constants to allow scalable tuning without need for logic changes
*MaxTrials = **8** (called MaxTrialsPerGame)*
MaxLevel = 100 (called MaxLevelBuckets)
*MaxScore = **65000** (Was 50000)*
MinTrialScore = 200
TrialScoreIncreaseByLevel = ((MaxScore - MinTrialScore) / MaxTrials) / (MaxLevel-1)
PercentByLevel = .1
PercentByPerformance = .6
PercentBySpeed = .3
*PercentRetry = **.5** (Was .3)*

PercentHintsUsed = .6
MinTime = 3000 (called minTimeScore)
MaxTime = 10000 (called maxTimeScore)
ExtraTimeByLevel = 25000 / MaxLevel (in milliseconds)

//Calculate TrialScore After a Trial, starting by calculating max possible points and reductions
*MaxTrialScore = MinTrialScore + (Level-1) * TrialScoreIncreaseByLevel*
PartialCreditRedux = (Tries > 1) ? PercentRetry : 1
HintRedux = (Hints < 1) ? 1: PercentHintsUsed

//Calculate a 0 to 1 ratio for speed, 1 if at or below the fastest time, 0 if above the slowest
***SpeedRatio** = (Response < MinTime) ? 1: ((Response > (MaxTime+Level*ExtraTimeByLevel) ? 0: MinTime / Response*

*TimeBonusScore = MaxTrialScore * PercentBySpeed * SpeedRatio;*

***TrialScore** = MaxTrialScore * (PercentByLevel + PartialCreditRedux * HintRedux * (PercentByPerformance + PercentBySpeed * SpeedRatio)*

//Rounding is assumed for final score as done in other games>

Progression Formula

Progress is determined by performance, momentum (the number of correct or incorrect answers in a row), speed and penalized when users use hints or require a retry to correctly solve a question.

Level advancement from trial to trial has been scaled to allow perfect players to reach a high peak in about 4-5 sessions. This system accelerates advancement when players answer multiple trials correct (1st try) in a row, with forward advancement occurring at twice the rate of possible regression.

*MaxAdvance = MaxLevel[currently:400] / (MaxTrials [currently:8] * 3)*
*SpeedBonus = (is Fast [currently:6000ms]) ? .15 * MaxAdvance : 0*
*HintPenalty = (Hints == 0) ? 0 : .25 * MaxAdvance*
*RetryPenalty = (FailedTries == 0) ? 0 : .45 * MaxAdvance * FailedTries*
*MomentumScalar = (consecutivePerfects > 3) ? 1 : (.2 + .2 * consecutivePerfects)*
*nextLevel += Round (MomentumScalar * MaxAdvance + SpeedBonus - HintPenalty - RetryPenalty)*

// MaxLevel would need to be > 50 to insure the desired graduation
// Perfect plays are those correct on the 1st try (with or without hints)
// Advancement rounds to a whole number

Trial Advancement - expected results from the algorithm are shown below:

		Last Trial Not Perfect	Last Trial Perfect	Last 2 Trials Perfect	Last 3 Trials Perfect	Last 4 Trials Perfect
Fast	Right 1st Try without hints	1	2	3	4	5
	Right 1st Try without hints	1	2	3	3	4
Fast	Right 1st Try with hints	0	1	2	3	4
	Right 1st Try with hints	0	1	1	2	3
Fast	Right 2nd Try without hints	0	0	1	2	3
	Right 2nd Try without hints	-1	0	1	1	2
Fast	Right 2nd Try with hints	-1	-1	0	1	2
	Right 2nd Try with hints	-2	-1	0	0	1
	Wrong Both Tries no hints	-3	-2	-1	0	0
	Wrong Both Tries with hints	-4	-3	-2	-1	-1

Metadata

Various factors contribute to the overall difficulty of a level, and metadata should provide enough info that we can determine the following (directly from metadata or indirectly via a puzzle configuration look-up table):

Meta-data structure

game_result

- (should be filled out as normal for games…)

trial_csv

- start_level
- end_level
- points
- response_time
- num_errors - incorrect submissions [int]
- difficulty - index in trial library
- ratio_1_type - Most difficult probability concept included (Enum.Concept) [[int]]
- ratio_2_type - Most difficult probability concept included (Enum.Concept) [[int]]
- numeratorRatio1 [pref empty, or 1/1 doesnt matter]
- denomRation1
- numeratorRation2
- denomRation2
- num_locks [int]
- locked_items - ; delimited list identifying starting objects (color|pattern|shape) use PIPES
- num_attempt [int]
- num_hints [num times hit hint button] [int]

- par - minimum placements necessary to correctly answer

response_csv

- attempt
    - Int, 1 if first attempt, 2 if second attempt
- correct
    -
- time_offset
- num_placements
- num_deletes - tracks items manually removed (ignoring clears)
- num_clears - tracks number of times user cleared entire board

action_csv

- Trial_id - out of the 8 total trials in a session [int]
- type - what type of action did the user attempt
    - object type toggled - when user switches the type of object they want to add
    - object added
    - object deleted
    - cleared
    - submitted
- Time_offset - use time_offset
- valid - T or F, was the action successful (valid and allowed) or refused (invalid) [nice to have]
- position_x
- position_y

Question Types

There are about 20 different types of probability questions being asked (see the difficulty spreadsheet), with each type being defined by a combination of a desired outcome (aka “Left Ratio Type” in the spreadsheet) a comparator (aka the operator) and a desired probability (aka “Right Ratio Type”).

Questions ask players to set the game up so that the chance of getting...

[Desired Outcome] is [Desired Probability]

...where [Desired Outcome] specifies a [Color] [Object Type] combo and can be a

- Simple Outcome: [Outcome] a Blue Bear
- Either-Or Outcome: [Outcome A] or [Outcome B] a Blue Bear or a Yellow Bear
- Sequenced Outcome: [Outcome] [Sequence Qualifier] a Blue Bear Twice in a Row

...and where [Desired Probability] can be

- Simple Probablility: [Probability] in following forms:
  - Fractional Probability: [Fraction] 4 / 5...
- Relative Probability: [Comparator] the chance of getting [Desired Outcome 2]
  - Equality Comparator: [=] equal to
  - Relative Inequality: [< >] greater than, less than
  - Magnification Comparator: [Magnification] 2x, 3x
  - Reduction Comparator: [Reduction] 1 / 2, 1 / 3, 1 / 4