Quick, Pass Me the Optimal Amusement Park Queuing System!

COMAP Competition
5-9 February 2004

Abstract

Queuing systems currently in place in amusement parks are inherently deficient. They frustrate park attendees, who have a list of complaints. We propose a superior queuing system, QuickPass. Through our mathematical analysis, we optimize the queuing system given a set of constraints, and all parties involved benefit. We find that placing a virtual gap between QuickPass time windows allows the greatest number of QuickPasses to be issued during a given day, which yields the lowest aggregate wait time. In addition, our model addresses user complaints, increasing overall guest satisfaction. Furthermore, our model will generate a significant increase in park revenue.

Table of Contents

1. Introduction

1.1 Raison d'Être: Amusement Parks

While waiting in a long line for a new, extreme roller coaster, standing in the oppressive summer heat, amidst a hoard of children with stuffed animals larger than themselves, the writers of this paper once posed the rhetorical question, Why would anyone *want* to go to an amusement park? Certainly, numerous reasons exist. Maybe you love roller coasters. Perhaps you go to a theme park or resort, such as Walt Disney World®, for a short vacation. Maybe you want to take your wife and 2.4 children out for a day of diversion. Whatever the particular reason, the common thread among these is that you want to experience pleasure and/or relaxation. You want to *enjoy* yourself. Perhaps your life is filled with unpleasantries, both grave and trifling. If you're Lester Burnham from *American Beauty*, you "feel like [you've] been in a coma for the past twenty years. And [you're] just now waking up." That's what an amusement park can be – an escape from the ordinary.

Thus, the best amusement park in theory, in contrast with Plato's best city in speech, produces the greatest enjoyment possible for its guests. This can be viewed as enhancing the pleasurable aspects of the park and reducing the undesirable elements. One such undesirable aspect of an amusement park experience is the long lines. Amusement parks are experimenting with various forms of queuing systems that allow riders to take a ticket that instructs them to return at a later time, at which point they can board the ride with little or no time spent in line. In this paper, we refer to such a queuing system as QuickPass (QP). QuickPasses issued from a kiosk include, at the least, a return time, which is the time at which the QP holder should return to ride. QPs may contain other information, as we will later show.

There are several potential models for a queuing system; however, some do not increase the enjoyment of the guest. One such model would be a stack-based approach; in other words, last-in-first-out (LIFO). With this model, the first people that get in line will be the last ones to get on the ride. The last people to enter the line would be the first ones to ride. Overall, the total aggregate wait time is the same as a queue-based model (first-in-first-out; FIFO). However, we can all agree that this model is absurd. Even though the wait time for the last people in line will be tiny, and the wait time for the folks in the middle will be about the same, the problem lies with the first people to enter the line – they won't ride until the line empties out! This brings up an interesting point: this LIFO system would most likely cause the situation to occur where there are no lines! However, not many people would ride the ride, park attendance would diminish, and the park would lose lots of money. Without loss of generality (WLOG), we can throw the stack-based approach out the window with the daily garbage.

For an in-depth look at a variety of models that we have discounted, see below in Section 5, The Model, Part I under constraints. There, we discuss varying extreme values of how many QP's to issue per window, a "train schedule"-like model, and more. These models are rejected precisely because they do not enhance the enjoyment of park attendees, which we determine to be a central goal of the QuickPass system.

1.2 Complaints

While minimizing time spent waiting in line is important, it is not the only concern in maximizing people's enjoyment of the amusement park. A QuickPass system will impact the park attendees' senses of fairness and just desserts. Thus, in order to understand how best to optimize a person's amusement park experience, we considered it important to address various complaints that guests have offered about QuickPass-like systems already in place. On the allearsnet.com website, Julie offered these comments about FastPass, a system that Disneyland® has implemented:

> We have mixed feelings about the FastPass system. When you get to a ride that has FastPass and look at the two clocks remember that the standby line time is an ESTIMATE of how much time you will need to wait to ride. We found that on some rides the time was way off and the line was much shorter that the clock said. We also found that on some rides the standby time estimate was too short because a lot of fast pass riders came in and their line backed up. The CM let the FastPass riders on and left us standing in line even longer when this happened. It was frustrating to be in the standby line for longer than the time estimate said on the clock and have the FastPass riders cut in front of you. FastPass made it hard to predict how long the wait for a ride would REALLY be.
>
> We found that FastPass changed the way we toured the parks. We went to the ride in each park that we really wanted first. If the stand by line was not very long we would get a FastPass first and then get in the standby line and ride it. By the time we got through riding it would be time for our FastPass. This let us ride 2 times in a row early in the day. FastPass did not work as well for us as the parks got more crowded. We would get a FastPass when we could for a ride, but the return times were usually a couple hours away. We found ourselves waiting around for a FastPass time to come up rather than moving to a different area of the park or walking more to go back to a ride that we were holding a FastPass for. I'm not sure if this system is worth it. Maybe it was better when everyone had equal rights in line?[1]

We will now address this guest's complaints.

1.2.1 More Accurate Estimates

Using our system, there will always be an accurate range of waiting times on a clock that a guest will be able to see at the beginning of the regular line. In our model, the accuracy of this range depends on being able to track how many people are in the line at any given time, which is not difficult to ascertain. As a result of this improved estimation system, there will never be the problem of having a displayed wait time of 90 minutes when there's really just 4 people in line, nor will the wait time of people in the regular line increase beyond the upper range of wait time listed on the clock, even if a "burst" of QuickPass holders show up during that time. This will be discussed in Section 7, The Model, Part II.

---

[1] "FastPass." AllEarsNet.

1.2.2 QuickPass holders will never fill a ride

Additionally, every line must always be moving. A person standing in the front of the regular line, which is not moving, while guests in a QuickPass line continue to cut in front of him, will not be a happy camper. In our model, whenever a horde of QuickPass riders show up, the proportion of QuickPass riders accepted on each ride will increase to prevent their line from backing up. However, we will still allow a certain percentage of people in the regular line on each ride so that both lines continue to move. Thus, although the rate at which the regular line moves will slow during a QuickPass "burst" of riders, overall it will reduce the total waiting time that everyone experiences. After the group of QuickPass riders is taken care of, the proportion of guests in the regular line who get to ride will increase again. We will discuss this constraint in Section 5.

1.2.3 Maximum Allowable Wait Time for QuickPass Holders

In our model, we allow the amusement park to choose a maximum acceptable waiting time for people in the QuickPass line. The number of QuickPasses and the duration for the time windows for each ride are based on this maximum time in such a way that a person in the back of the QuickPass line will never spend more than the set maximum time in line. This is very important, as QP holders will not tolerate much of a wait when they return at their scheduled time, even if their total time spent in line is far less than it would be otherwise. A QuickPass is something of a contract, promising its holder little or no wait time in line, in exchange for agreeing to return at a later time. The inconvenience of returning later is rewarded by a greatly reduced wait, and if the park does not fulfill its promise of a short wait for QP holders, it is effectively reneging on its contract. The QuickPass holder will feel that he is being deprived of something he rightly deserves, even though in terms of time spent waiting in line, he is better off.

1.2.4 QuickPass Return Times & Available QuickPass Windows

Because there are only a limited number of QuickPasses that can be allocated in each window, there is no choice but to give QuickPasses scheduled later and later in the day as more people ask for them. So if someone really wants to get on a particular ride, he has two choices: they can go ahead and wait in the regular line, which will take no longer than it would if the QuickPass system didn't exist, or they can take a QuickPass for later on in the day and enjoy other aspects of the amusement park in the meantime. It is up to each guest to determine if a QuickPass provides more benefit by decreasing the amount of time spent waiting in line, or more cost by having an imposed schedule to follow.

One aspect of our model that will decrease the rigidity of a guest's schedule is the capability to choose which open time window for which he would like the QuickPass. So, for example, if he wanted to ride as soon as possible, he could still choose the next available time window, but if he wanted to visit a different area of the park for a period of time, he could choose a later time and avoid having to stay in his current area waiting for his window to open up.

1.2.5 All Men are *Created Equal*[1]

One thing that guests must keep in mind is that the QuickPass system we are using is minimizing the aggregate wait time. Using our model, nobody is ever worse off than they would be if the system didn't exist, while the majority of people benefit. To someone

[1] Declaration of Independence, 1776.

standing in the regular line watching many QuickPass riders pass in front of them, this may seem hard to accept. However, these passes are available to everyone in the park, so that same disgruntled guest can later acquire a QuickPass for himself and end up on the opposite side of that situation. Furthermore, we will demonstrate that our QuickPass system is Pareto optimal, i.e. that everyone is at least as well off, and some are better off.

1.2.6 Riding Back-to-Back

One option this QuickPass system provides is the capability to ride a very popular ride two times in quick succession, which Julie seemed to enjoy. Once obtaining a QuickPass with a given return time, a guest's free time can be spent standing in line for the ride anyway. This way, some period of time after the guest gets on the ride through the regular line, his time window for the QuickPass line will open up. By doing this, his presence in the regular line will cause everyone behind him to wait longer, since the whole idea of the QuickPass is that the people who have one will not be in the line until their window opens. That said, many guests enjoy this feature of QuickPass systems, and since they are available to everyone, anybody can double up on their favorite ride. Our model allows for this possibility since we require that the return time for a QuickPass is always at least as long as the current wait time in the regular line.

The problem we face is, on the one hand, to improve the actual operation of the QuickPass system and, on the other hand, to increase the enjoyment of folks attending the park. Furthermore, we will consider the potential for the amusement park's financial through an appropriately designed QuickPass system. Reduced to its simplest elements, our task is to maximize enjoyment and revenue, given a set of constraints. Our model will do just that – and it is legal in all 50 states. Now, please sit back, keep all hands and loose objects inside the car at all times, and enjoy the ride as we introduce the mathematics behind our model.

2. General Assumptions

The following have been assumed unless temporarily revoked elsewhere in the paper.

We assume that the greatest good for the individual is as important as the greatest good for the community.

We assume that the given ride will always be running at full capacity. This is reasonable to assume since the QuickPass system is designed for rides that are in high demand. Thus, there should be enough people to fill the ride every time.

We assume a normal, functioning operating day (i.e., the ride never breaks down).

We assume that the rain in Spain falls mainly on the plain (except when it falls in Hartford Haverford, or Hampshire).

We assume that a QuickPass Kiosk will be able to obtain data, such as the current time, the current number of people standing in a given line, etc.

We assume that the amusement park will never want to issue a QuickPass for a window of time that is sooner than the current wait time for the regular line.

In our system, we define QuickPass windows as non-overlapping (e.g., if one window is from 4:30-5:15pm, a window from 5-5:45pm will not exist).

3. Notation

The variables used in this paper are defined below for the benefit of the reader. They are ordered alphabetically, more or less.

δ_{cur} = Current time

δ_{day} = Time of day that the return time specifies, $\delta_{day} = \tau_{cur} + \delta_{ret}$

h_{ret} = "return time," i.e. length of time from the current time until a QuickPass tells a rider to return (hours)

k = Capacity of the ride (people/ride)

λ = Wait time for the regular line (minutes)

N_{day} = Total number of QuickPasses issued during one day

N_{fast} = Number of people from the regular line boarding the ride at the fastest possible rate

N_{line} = Number of people in the regular line

N_{QP} = Maximum number of QuickPasses to be issued in one window

N_{reg} = Number of riders admitted from the regular line during one window

$N_{reg_{passed}}$ = Number of people from the regular line who have ridden during the current window

N_{slow} = Number of people from the regular line boarding the ride at the slowest possible rate

$N_{tot} = N_{reg} + N_{QP}$ = Capacity of the ride during one window

ϕ_{avg} = Average proportion of QuickPass riders allowed per ride, $0 \le \phi < 1$

$\phi_{\max}$ = Maximum proportion of QuickPass riders allowed per ride, $0 \le \phi_{\max} < 1$

ψ = Depreciation constant of how many QP holders will not show up during a window that occurs later in the day

r = Inverse of the frequency of the ride (minutes/ride)

τ_{cur} = Current time, converted to minutes

τ_{day} = Total time the park is open (minutes)

τ_g = Length of the virtual gap (minutes)

τ_ℓ = Maximum acceptable wait time for people in the QuickPass line (minutes)

τ_{open} = Amusement park opening time, converted to minutes

τ_W = Length of window for QuickPasses (minutes)

W = Total number of windows in a given day

4. Equations

The following are the principal equations used in this paper. All derivations, explanations, clarifications, elucidations, and illuminations can be found within the succeeding pages.

$$N_{tot} = N_{reg} + N_{QP} = \frac{k}{r}\tau_W$$

$$N_{QP} = \frac{\phi_{avg} k}{r}\tau_W = \frac{1}{\upsilon}\frac{\phi_{\max} k}{r}\tau_\ell$$

$$N_{reg} = \frac{k}{r}\tau_W - \frac{1}{\upsilon}\frac{\phi_{\max} k}{r}\tau_\ell$$

$$\phi_{\max}\tau_\ell = \upsilon\phi_{avg}\tau_W$$

$$\upsilon = \frac{2\tau_\ell - \tau_g}{\tau_\ell}$$

$$N_{day} = \frac{k}{r}\phi_{avg}\tau_{day}$$

$$\lambda_a = \tau_W\left\lceil \frac{\tau_{cur} - \tau_{open}}{\tau_W} \right\rceil - (\tau_{cur} - \tau_{open})$$

$$\lambda_b = \tau_W\left\lfloor \frac{N_{line} - (N_{reg} - N_{reg_{passed}})}{N_{reg}} \right\rfloor$$

$$\lambda_{a+b} \equiv \lambda_a + \lambda_b = \tau_W\left(\left\lceil \frac{\tau_{cur} - \tau_{open}}{\tau_W} \right\rceil + \left\lfloor \frac{N_{line} - (N_{reg} - N_{reg_{passed}})}{N_{reg}} \right\rfloor\right) - (\tau_{cur} - \tau_{open})$$

$$N_{slow} = \frac{(1-\phi_{\max})k}{r}\tau_\ell$$

$$N_{fast} = \frac{k}{r}(\tau_W - \tau_\ell)$$

$$\lambda_{\max} = \lambda_{a+b} + p\left(\frac{r}{(1-\phi_{\max})k} N_{left}\right) + q\left(\phi_{\max}\tau_{\ell} + \frac{r}{k} N_{left}\right)$$

$$if\ N_{left} \le N_{slow}, p = 1\ and\ q = 0$$

$$if\ N_{left} > N_{slow}, p = 0\ and\ q = 1$$

$$\lambda_{\min} = \lambda_{a+b} + p\left(\frac{r}{k} N_{left}\right) + q\left(\tau_W - \tau_{\ell} + \frac{r}{(1-\phi_{\max})k} N_{left} - \frac{\tau_W - \tau_{\ell}}{1-\phi_{\max}}\right)$$

$$if\ N_{left} \le N_{fast}, p = 1\ and\ q = 0$$

$$if\ N_{left} > N_{fast}, p = 0\ and\ q = 1$$

$$\lambda_{avg} = \lambda_{a+b} + \tau_W \frac{N_{left}}{N_{reg}}$$

$$\psi = \alpha h_{ret} + \beta \delta_{day} + 1$$

5. The Model, Part I

This section will derive the fundamental ideas behind the first aspect of our model. The QuickPass system is based on the idea of giving park attendees the choice between waiting in line for a ride now or returning later and waiting in a shorter line. For those who choose to return later, a kiosk will issue them a QuickPass, designating a time window in the future during which they can return to ride. They enter through a separate, much shorter line and ride.

First, we will derive the basic equations for the number of QuickPass users and regular users that a ride can accommodate during a window, then we will discuss realistic constraints which must be placed on our variables. Next we consider the possibility of adding a gap between the scheduled windows during the day, first discussing an example and then generalizing our model to include this gap time. Finally, we will discuss how to maximize the total number of QuickPasses handed out during a day, which, as we will see later, will minimize the aggregate wait time of the guests in the park. In effect, we approach this problem as constrained optimization.

5.1 Derivation of N_{QP} and N_{reg}

Beginning with the capacity of each individual ride k (people/ride) and dividing it by the duration of each ride r (minutes per ride), we obtain the number of people per minute that the ride can handle:

$$\frac{k}{r}$$

If we multiply this by the length of the QuickPass window, τ_W, we will obtain the total capacity N_{tot} of the ride for that window:

$$N_{tot} = \frac{k}{r}\tau_W \qquad \textit{(eq. 1)}$$

Multiplying N_{tot} by the total number of windows throughout the day, called W, let's say, will yield the daily capacity of the ride, which we must strive to fill to the fullest.

Now, if we simply multiply N_{tot} by ϕ_{avg}, the average proportion of QuickPass users let on each ride, then we will obtain the number of QuickPasses to issue during one window:

$$N_{QP} = \frac{\phi_{avg} k}{r}\tau_W \qquad \textit{(eq. 2)}$$

Similarly, if we multiply N_{tot} by $1-\phi_{avg}$, we will obtain N_{reg}, the number of people in the regular line that a ride can accept during a window:

$$N_{reg} = \frac{(1-\phi_{avg})k}{r}\tau_W$$

5.1.1 Derivation of Day-long formulas

Let us call the total number of QuickPasses issued during a day N_{day}, and the total time the park is open τ_{day}. Then the total time the park is open equals the total number of windows, *W*, times the length of each window:

$$\tau_{day} = W\tau_W$$

Similarly, the total number of QuickPasses issued during the day equals the number issued per window times the number of windows:

$$N_{day} = WN_{QP}$$

5.2 Constraints

5.2.1 Full capacity

Since the rides using the QP system are typically popular rides with high traffic and long lines, operating the machines at full capacity, or as full as humanly possible, should be a goal. This will allow as many park attendees as possible to ride the ride, thus increasing overall customer satisfaction.

5.2.2 Length of the Window: τ_W

The length of the window must have an established minimum value related to park size and inconvenience to riders. τ_W must be long enough so that people have sufficient time to cross from the other side of the amusement park at their leisure. If τ_W is too small, riders will be under pressure to return at a specific time, which takes away from their enjoyment at the park. Park attendees should be able to follow a flexible vacation schedule, not a regimented work agenda.

5.2.3 Average Percentage of QuickPasses to Issue: ϕ_{avg}

In first approaching the problem of finding the optimal percentage of QuickPasses to issue, we consider the extremes. First, what if $\phi_{avg} = 0$? This is the *status quo ante*, that is, there is no QP system in place. As we can show that a QP system can reduce aggregate wait time, we know that the optimal ϕ_{avg} is greater than 0.

On the other hand, $\phi_{avg} = 1$ means that every seat on every ride is reserved for QP users. If it worked ideally, there would be no waiting in line: all riders would show up for their designated time and ride immediately. It follows that at $\phi_{avg} = 1$, there should be no regular line. By definition, all seats on all rides for the day are assigned to someone with a QP; thus, there can be no regular line. Anyone who gets on the ride and does not hold a QP will deprive a QP holder of a seat.

Aside from the improbability of perfectly punctual humans, letting $\phi_{avg} = 1$ has two consequences, both of which contradict one of the two constraints listed above. One is that, in order to ensure the fullest capacity of the ride, we would have to set $\tau_W = r$. This means that every QP user is issued a ticket for a specific ride, as in a precise train schedule, and it clearly contradicts the restriction on τ_W. It would make navigating an amusement park even

more stressful than it already can be (due to large crowds, screaming children, people dressed as giant cartoon characters, the sensual, yet pure, beauty of the unattainable Snow White, etc.).

Even setting $\tau_W = r$, however, would not ensure that the ride is full every time. What if someone misses their QP time? Under lower values of ϕ_{avg}, the empty seats would be filled by people from the regular line, but we have no regular line at $\phi_{avg} = 1$. If a de facto regular line forms, people could be admitted from it, but there is the risk that QP holders will return and be unable to ride. Wait times for the QP line will rise, some QP holders will not get to ride despite the implicit guarantee that the QP provides. The entire system will be undermined.

Increasing τ_W to a value acceptable within the constraint will not solve the problem of empty seats. With larger values of τ_W, the chance that QP holders will miss their scheduled window decreases, but an even distribution of QP holders showing up to ride throughout is unlikely. Empty seats will still result.

$\phi_{avg} = 1$ has other consequences as well. Since a popular ride only has a limited daily capacity of people it can accommodate, only that many QPs could be given out. At $\phi_{avg} = 1$, park attendees who arrive early could reserve seats on a ride for the whole day, while a family who shows up at 2:00 PM would arrive to find that there are no more QPs left, thus prohibiting them from enjoying the ride for the rest of the day. Also, this would promote extremely long return times, scheduling the person who picks up the last QP for a popular ride in the morning for a time 5 minutes before closing. This method does not seem to be a practical way to queue people. The bottom line is that since we need a regular line to ensure a full ride whenever possible, ϕ_{avg} must be less than 1. We will revisit ϕ_{avg} later.

5.2.4 Maximum Wait Time for QuickPass Holders in Line: τ_ℓ

The QP system is designed to reduce wait times, yet it brings with it the inconvenience of delayed gratification. Thus, the wait time in the QP line, once a person has returned during his assigned window, must be significantly lower. One user complaint of current QP-like systems is that they have still had to wait in line for a while even when using the QP. To remedy this problem, we suggest setting a maximum wait time for QP users. This maximum wait for QP riders, τ_ℓ, should be presented as a guarantee to park attendees, and adhering to it – as this model ensures – will increase customer satisfaction.

One restriction we will have to impose is that $\tau_\ell \geq r$. Although QP holders might appreciate $\tau_\ell = 0$, we can never guarantee that a QP user will have no wait. Inevitably, users will show up while the current ride is still going, so that at least a brief wait time will be experienced until the next ride departs.

Another point about τ_ℓ is that it is one factor determining the largest possible burst of QP riders, meaning the maximum number of QP holders that can show up at one time. The last person in the QP line (who theoretically arrived at the exact same time as everyone else in front of him, yet ended up last in the queue) must be able to ride in τ_ℓ minutes.

5.2.5 Maximum Percentage of QuickPasses to Issue: $\phi_{\max}$

In order to ensure that no QuickPass user will wait longer than τ_ℓ, the proportion of QuickPass users let on each ride during peak hours must be allowed to increase to an upper limit, $\phi_{\max}$. Thus, the maximum rate at which a ride can process QuickPass users is $\frac{\phi_{\max} k}{r}$. As a proportional constant, $\phi_{\max}$ cannot be greater than 1. However, we argue that $\phi_{\max}$ must be significantly less than 1.

During a burst of QP riders, it seems that the best way to expedite the QP line would be to increase ϕ all the way to 1. However, one must consider the effect this has on the front of the regular line when the QP burst arrives. If $\phi_{\max}=1$, then the front person in the regular line must wait there until all of the QP users have been ridden. This would obviously anger the people in front of the regular line. For example, recall part of Julie's complaint mentioned earlier:

> We also found that on some rides the standby time estimate was to short because a lot of FastPass riders came in and their line backed up. The CM let the FastPass riders on and left us standing in line even longer when this happened. It was frustrating to be in the standby line for longer than the time estimate said on the clock and have the FastPass riders cut in front of you.

Hence, we will place an upper limit on ϕ less than 1 so that both lines will always be moving. This will reduce frustration of users in the regular line.

5.3 The Gap Extension

Following from the discussion of τ_ℓ and $\phi_{\max}$, we should consider what factors determine the largest possible burst of QP holders. We will show that placing a virtual gap in between windows will reduce or eliminate the problem of bursts from consecutive windows overlapping. This will allow a larger number of QPs to be issued, given the other constraints, and more QPs equate to less aggregate wait time.

First, let us discuss the case when there is no gap between windows. The worst case, i.e. the largest possible burst, would be if all of the QP holders from one window showed up at the end of the window, and then all of the QP holders for the following window showed up at the beginning of the window. For example, all QP holders for windows from 8:00 to 8:45 and 8:45 to 9:30 show up at 8:45. In this case, the ride has τ_ℓ to accommodate the QP holders from both windows, at a rate of $\frac{\phi_{\max} k}{r}$; hence,

$$N_{QP} = \frac{\phi_{\max} k}{r} \tau_\ell \qquad \textit{(eq. 3)}$$

But we already know from equation (2) that for any window, $N_{QP} = \frac{\phi_{avg} k}{r} \tau_W$, and since in this scenario the ride is processing all of the QuickPass users from both windows at once, we have

$$N_{QP} = \frac{2\phi_{avg} k}{r} \tau_W \qquad \textit{(eq. 4)}$$

Equating equations (3) and (4) and simplifying, we find the relation:

$$\phi_{\max}\tau_\ell = 2\phi_{avg}\tau_W$$

Now, let us consider what happens if we insert a virtual gap, τ_g, between windows. This means that the QuickPass will tell riders to return to the ride during a window that is smaller than the actual window. For example, for windows from 8:00 to 8:45 and 8:45 to 9:30, the kiosk might report windows of 8:00 to 8:30 and 8:45 to 9:15, respectively. In this example, a gap of 15 minutes is built in. Note that our earlier constraint for the minimum time of τ_W should really be applied to $\tau_W - \tau_g$. We want to allow guests enough time to conveniently reach a ride within the window they see on the QuickPass (for the example above, 30 minutes). If we set $\tau_g = \tau_\ell$ =15 minutes, even if all QP holders show up at the end of their reported window, they will all be able to ride before the next window starts. Thus, there can be no compounding of QP bursts between windows. Now our expression for N_{QP} will only need to account for one window instead of two:

$$N_{QP} = \frac{\phi_{avg}k}{r}\tau_W \qquad \textit{(eq. 5)}$$

Again, equating (3) and (5) we now get the relation:

$$\phi_{\max}\tau_\ell = \phi_{avg}\tau_W$$

Hence, there seems to be a dependence among these variables on the gap time. Suppose that we are given two windows of duration τ_W, each with a gap of τ_g after it. Also, suppose that all of the QuickPass users in the first window show up at the end of their allowed time, and after the gap, all of the QuickPass users from the second window immediately show up as well. The ratio of QuickPass users who will carry over to the second window will be $\frac{\tau_\ell - \tau_g}{\tau_\ell}$, so the total number of QuickPass users who must ride in the second window can be written more generally as:

$$N_{QP} = (1 + \frac{\tau_\ell - \tau_g}{\tau_\ell})\frac{\phi_{avg}k}{r}\tau_W$$

$$N_{QP} = (\frac{2\tau_\ell - \tau_g}{\tau_\ell})\frac{\phi_{avg}k}{r}\tau_W \qquad \textit{(eq. 6)}$$

Call this new proportion in front υ:

$$\upsilon = \frac{2\tau_\ell - \tau_g}{\tau_\ell} \qquad \textit{(eq. 7)}$$

Fortunately, we have already established that $\tau_\ell \neq 0$. We don't want any pesky singularities wandering around our amusement parks.

5.3.1 Restrictions on τ_g

It seems appropriate here to take an aside and establish some constraints on values that τ_g can take. The length of this gap, if it exists, should at least be as long as the duration of our ride, *r*, because otherwise it will not reduce the wait time of any QuickPass holders who arrive at the end of a window. For example, suppose the duration of a ride is 3 minutes. If the gap time is 1 minute and 20 people show up at the end of an window, then after the gap is over, they will still be waiting to board the ride when the QuickPass users for the next window arrive. More generally, a gap will be most beneficial if it is a multiple of the duration of the ride. Hence, we will say that:

$$\tau_g = cr : c = 0,1,2,...$$

Furthermore, it does not make sense to allow the gap time to be greater than the maximum allowable wait time, τ_ℓ. The purpose of the gap is to allow some of the QuickPass users who arrive late in a window to be processed before the next window opens up. So if a gap lasts longer than the greatest possible wait time, there will be time where no QuickPass users will be able to show up, and this is inefficient because it reduces the total number of QuickPass users that the ride will accommodate in a day, which we are trying to maximize. Thus, $\tau_g \le \tau_\ell$.

Returning to our expressions (3) and (6) for N_{QP} in terms of ϕ_{avg} and $\phi_{\max}$, we equate them and find a general relation among our variables:

$$\phi_{\max}\tau_\ell = \upsilon\phi_{avg}\tau_W \qquad \textit{(eq. 8)}$$

This lets us create two equivalent expressions for N_{QP}:

$$\boxed{N_{QP} = \frac{\phi_{avg}k}{r}\tau_W = \frac{1}{\upsilon}\frac{\phi_{\max}k}{r}\tau_\ell} \qquad \textit{(eq. 9)}$$

Let us finally return to N_{reg} and show some alternative ways of representing it:

$$N_{reg} = \frac{(1-\phi_{avg})k}{r}\tau_W$$

$$N_{reg} = \frac{k}{r}\tau_W - \frac{\phi_{avg}k}{r}\tau_W$$

$$N_{reg} = \frac{k}{r}\tau_W - N_{QP}$$

$$N_{reg} = \frac{k}{r}\tau_W - \frac{1}{\upsilon}\frac{\phi_{\max}k}{r}\tau_\ell \qquad \textit{(eq. 10)}$$

Note that these derivations are consistent, since if we add N_{QP} to both sides of this last equation, we recover the formula for N_{tot}:

$$N_{tot} = N_{reg} + N_{QP} = \frac{k}{r}\tau_W$$

5.4 Maximization of N_{day}

Given our derived equations (7) and (8)

$$\phi_{\max}\tau_{\ell} = \upsilon\phi_{avg}\tau_{W}$$

$$\text{where } \upsilon = \frac{2\tau_{\ell} - \tau_{g}}{\tau_{\ell}},$$

we can substitute υ into the first equation as follows:

$$\phi_{\max}\tau_{\ell} = (\frac{2\tau_{\ell} - \tau_{g}}{\tau_{\ell}})\phi_{avg}\tau_{W}$$

$$\frac{\phi_{\max}\tau_{\ell}^{2}}{2\tau_{\ell} - \tau_{g}} = \phi_{avg}\tau_{W} \qquad \textit{(eq. 11)}$$

First, let us explore the relationships among these different variables. τ_{ℓ} and $\phi_{\max}$ will be predetermined by the amusement park, based on the convenience they want to confer on their guests. The park is free to choose, within constraints, how small they want the maximum wait time, τ_{ℓ}, for QuickPass users to be, as well as what proportion of regular users, $1-\phi_{\max}$, are guaranteed to get on each ride so that people at the front of the regular line do not become frustrated. These variables being fixed, we are left with a relationship among τ_{g}, τ_{W}, and ϕ_{avg}. Recall the first part of equation (9) for the number of QuickPasses to be given out during one window:

$$N_{QP} = \frac{k}{r}\phi_{avg}\tau_{W}$$

For a given ride, *k* and *r* will be fixed, so the total number to QuickPasses handed out will be fully dependent on the product $\phi_{avg}\tau_{W}$, which also appears above. What we want is to maximize the number of QuickPasses given out during the day under our constraints. Doing so will decrease the aggregate wait time, which has several positive effects. Park attendees will spend less time waiting in line, which should increase the enjoyment that they experience at the amusement park. For park owners, the decrease in aggregate wait time will increase the amount of free time that park guests have, equating to more spending time from the park's perspective.

Recall our formulas earlier in the derivation for the total number of QuickPasses issued during a day N_{day}, the total time the park is open τ_{day}, and the total number of windows throughout the day W:

$$\tau_{day} = W\tau_{W}$$

Solving for W we have:

$$W = \frac{\tau_{day}}{\tau_{W}}$$

We also recall that:

$$N_{day} = WN_{QP}$$

We want to maximize N_{day}, so let us plug in what we know and find out what makes this guy tick:

$$N_{day} = \frac{\tau_{day}}{\tau_W}\left(\frac{k}{r}\phi_{avg}\tau_W\right)$$

$$N_{day} = \frac{k}{r}\phi_{avg}\tau_{day} \qquad \textit{(eq. 12)}$$

This equation reveals several interesting points. Increasing τ_W allows the park to increase the number of QuickPasses issued per window, yet it also reduces the total number of windows during the day. This equation shows that in fact, the countering forces of τ_W balance each other out exactly, so our maximization of N_{day} is independent of τ_W. Everything in the above equation for N_{day} will be fixed ahead of time except for ϕ_{avg}. So, that is the key: *to maximize the number of QuickPasses to be issued during the day, maximize the average number of QuickPass users allowed on each ride.*

Now, let's return to equation (11) relating τ_g, τ_W, and ϕ_{avg}:

$$\frac{\phi_{\max}\tau_\ell^2}{2\tau_\ell - \tau_g} = \phi_{avg}\tau_W$$

Since we know that τ_W will not play a role in the overall goal, and since everything else in this equation will held constant, we are left with the following relationship: an increase in τ_g corresponds to an increase in ϕ_{avg}, which is what we want. So, we would like to make the gap time as large as possible, because doing so directly implies that more QuickPasses can be issued throughout the day. Earlier, we gave evidence that the upper bound for τ_g should be equal to the maximum allowable wait time τ_ℓ, since a gap time larger than τ_ℓ would waste potential arrival time for QP users. Therefore, the main conclusion is that *setting $\tau_g = \tau_\ell$ will maximize the number of QuickPasses that can be issued during the day, thus decreasing the aggregate wait time experienced by everyone in the park.*

In the following section, we will discuss some extended examples with real numbers to demonstrate the conclusions found in this section: that having a QuickPass system at all is preferable to just using a regular line, and that having a gap between each window in a QuickPass system is preferable to not having a gap.

6. Simulated Data: Testing the Model

6.1 Mind the Gap. Mind the Gap. Mind the Gap. Mind the Gap. Stand clear of the doors, please.[3]

In this section, we will give an example that demonstrates that, subject to all of the constraints mentioned above, having a gap is better than not having a gap. This will help fortify the algebraic conclusions drawn earlier. Furthermore, having QuickPasses at all is better than not having them.

Before addressing the example, we should clarify what "better" means. In terms of utilitarianism, one way of evaluating two states or situations against each other is to compare their social welfare functions, or the sum of the utilities of every individual. There are several problems with doing this in the real world. However, in this case we are comparing states based on their aggregate time spent in line. Though we cannot assume that all people value time equally, we can make a strict utility comparison of the two states with greater facility since both states contain the same thing, namely time: we are not comparing Snow White to merely reading about Snow White.

On the other hand, a stricter method of comparing social welfare is Pareto optimality. For state A to be Pareto optimal to state B, all people must be at least as well off in state A as in state B, and at least one person must be better off in A than in B. Through the following example, we will show that having a virtual gap between windows not only reduces aggregate wait time, it is Pareto optimal to not having a gap. Both are Pareto optimal to the case in which no QuickPasses are issued.

Figure 1 shows these three cases given a set of sample values for the constants. For the actual numbers produced in this graph, we set the following values: $\tau_W = 45$, $\tau_\ell = 15$, $r = 3$, $k = 30$, and $\phi_{\max} = 2/3$. The time period in question is two full windows, as we are examining the case of the largest possible burst, i.e. when all of the QP holders from the first window show up at the end, and all of the QP holders from the second window show up at the beginning. Variables within the examples are ϕ_{avg} and the gap. In the graph, the black line represents individual wait times on the ride if no QuickPass system is in place. In this case, $\phi_{avg} = 0$, and the gap is irrelevant. The dark gray line depicts a QP system with no gap. Thus, $\tau_g = 0$ and, given our other values, $\phi_{avg} = 1/9$. The light gray line represents a QP system with $\tau_g = \tau_\ell$, and thus $\phi_{avg} = 2/9$.

Interpreting the meaning of the data points on the lines requires some explanation. Each diamond point represents 10 people. Consider the black line. There are three points corresponding to 0 on the *y*-axis, three at 3, three at 6, etc. This means that there are 30 people that wait 0 minutes, 30 that wait 3 minutes, 30 that wait 6 minutes, etc. The numbers on the *x*-axis aggregate the number of people, such that 30 on the *x*-axis corresponds with 0, 60 is 3, and 90 is 6. Thus, we can say that for any value of *x*, there are *x* people with a wait time of *y* or less. We cannot necessarily say, however, that the 90th person in line waits for 6 minutes. On the light and dark gray lines, QP riders are involved. After calculating individual wait times (using the Queuing Computer Simulation described below), the wait times for regular line people and QP line people were meshed together. In the dark gray

[3] As heard in the London Underground.

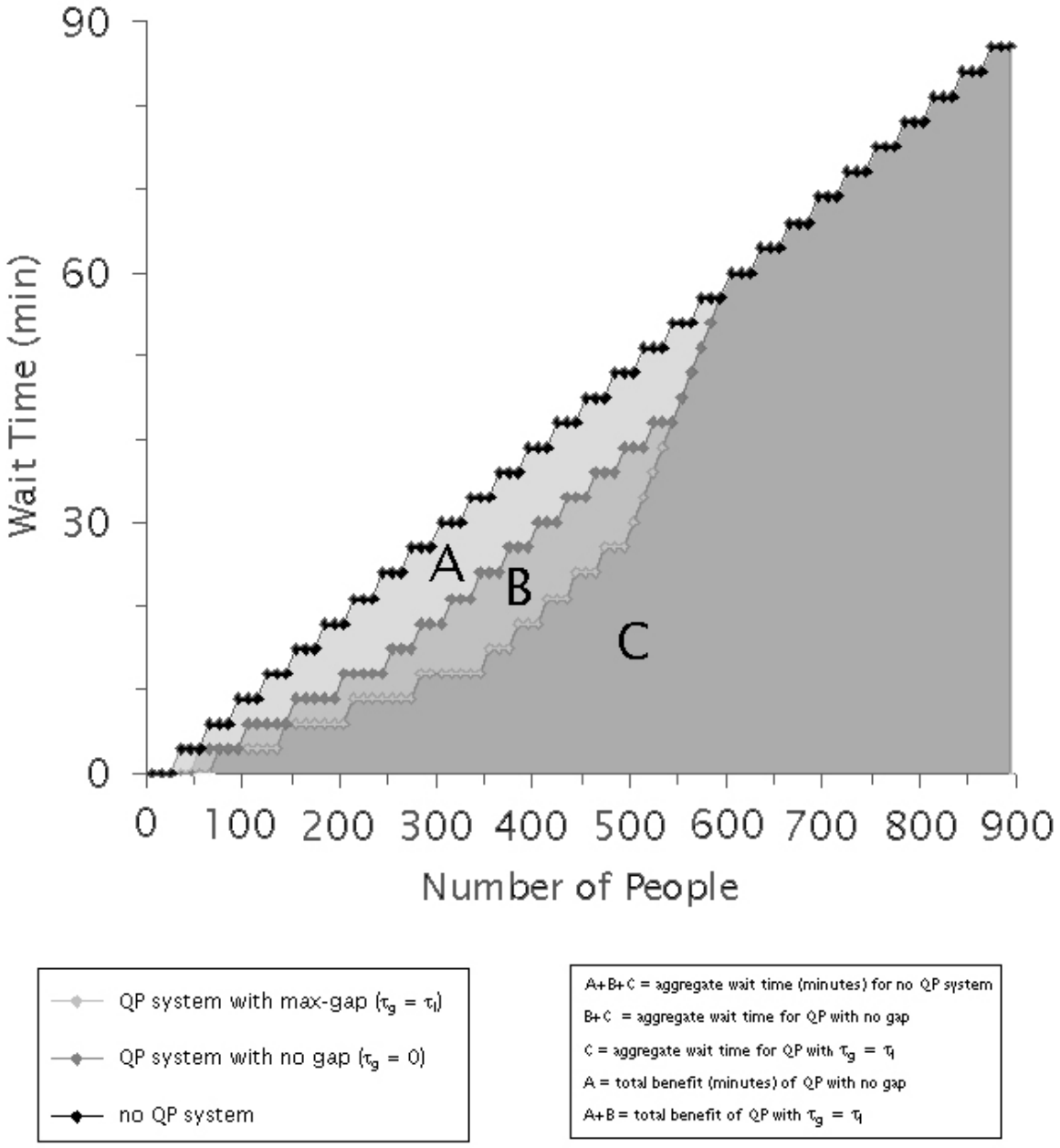

Figure 1: Wait Times & Aggregate Waits for Three Different Systems

line, for example, the first 30 people in the regular line ride with no wait, and the first 20 people (according to ϕ_{max}) in the QP burst can ride immediately as well. Thus, over the course of the period, 50 people ride with no wait. The points on the dark gray line represent this meshing of wait times.

The shaded areas under the curves represent aggregate wait times. The graph thus shows the logical negative correlation between the number of QuickPasses issued per

window and the aggregate wait time. Put otherwise, this means that *the greater the* ϕ_{avg}*, the greater the reduction of aggregate time spent waiting in line.* This is intuitive: the more people who are guaranteed a maximum wait time of τ_ℓ, the fewer that have to wait in a longer regular line. Our model holds even if τ_ℓ is greater than the wait time for the regular line. People will only take QuickPasses when the regular line is long enough that the inconvenience of returning later is rewarded with a shorter line. If the regular line is so short that it is less than the guaranteed maximum QP wait, the potential rider will simply enter the regular line and forego the QuickPass, all other things being equal. Thus, QuickPasses are designed with long regular lines in mind.

Note that wait times expressed in the graph are times spent waiting within this window, regardless of how much riders have already waited in line up to this window. As we are analyzing at the margin – and because we cannot know when people will show up at the ride – we consider the time spent in line before this window as a sunk cost and seek to reduce the wait time within this window. For simplicity in calculation, we assume that the regular line is effectively infinite within the scope of these two windows. That is, there are always enough people in the regular line to fill all seats on a given ride not filled by QP holders. Furthermore, there are more people in the regular line than can ride within the next two windows. We feel that this is a safe assumption because rides using the QuickPass system are rides that are in high demand. In the examples, we compare the minimized aggregate wait to the counterfactual case, i.e. in which there is no QuickPass system. Because of this comparison to the counterfactual 2-window time period, we argue that even if there are fewer people in the regular line than can ride within the next two windows, the difference in aggregate wait between the two cases will be unchanged, ceteris paribus.

In concrete terms, what does this reduction in aggregate wait time mean in terms of benefits? In our examples, area A, the time saved by a system with no gap, is 4,500 minutes, or 3,000 minutes per hour (since the two-window period is 90 minutes long). Area A + B, the time saved by a gap system, is 7,500 minutes, or 5,000 minutes per hour. For park attendees, the benefits are hard to quantify. The increased free time will allow them to do things other than waste the day standing in line. Certainly this will increase their enjoyment of their park experience – particularly if they have impatient young children.

The benefits accrued to the park, on the other hand, are very quantifiable. People only spend money in the park while walking around, not while standing in line. Thus, if we decrease the time spent in line, we increase the time that park guests have to do other things, one of which could be purchasing food or souvenirs. Thus, a QP system could directly increase the park's revenue. By using rough estimates of Walt Disney World's® annual number of visitors (40 million people) and how much each visitor is likely to spend in any given day ($52)[4] and given the time saved waiting in line, we estimate that Walt Disney World® would make an additional $193 million annually using this QuickPass system with a gap. This is about $77 million more than would be generated using the same QuickPass system without a gap. These figures, of course, correspond to the fictitious ride we imagined with a daily capacity given by *k* and *r*. Naturally, a larger ride would have a larger aggregate wait time, ceteris paribus, so a QuickPass system would save even more wait time, thus entailing a larger revenue boost for the park.

[4] Kirsner, Scott. "Hack the Magic."

Along the same reasoning, we can easily conclude, without loss of generality, that London's Underground, where one must "mind the gap," is superior to the New York subway system, where there are no gaps! They probably don't even use a QuickPass system!

6.2 Bursts: They Might Just Kill You…more on the 11 o'clock News

Using the same premises we set for the previous example for defending the gap,[5] we now set off to assess the impact of bursts of QuickPass riders on our system. Figure 2 is designed similarly to the former. The black line again depicts individual wait times for a system in which no QuickPasses are issued. The light gray line also depicts the same QP system as it does in the previous graph. The dark gray line represents, like the light gray line, a QuickPass system with $\tau_g = \tau_\ell$ and $\phi_{avg} = 2/9$. The difference is that instead of all QP holders arriving in a burst, they arrive evenly dispersed throughout the duration of the windows. Note that, once leaving the *x*-axis, it is effectively linear.

The graph shows that the aggregate wait time for the QP system with a burst and the aggregate wait time for the QP system with even distribution are not much different. That said, it appears that the difference between the light and dark gray lines is who benefits most. The dark gray line shows over 200 people with a wait time of 0. This includes all of the QP holders, who arrive at the appropriate moments in order not to wait, and the $1-\phi_{avg}$ proportion of the first ride in the period that is filled with regular line people. The more QP holders that arrive later in the period, the more regular line people that can ride earlier in the period.

Consider Table 1, which shows the corresponding wait times every tenth person in line faces for each of the three cases in Figure 2. Note that, unlike the way in which the graph data was sorted together, in the table the QP holders, shaded in gray, are listed at the beginning of each window. This is because the table is listing people by their order in line. As our QP kiosk system operates, as described later, QuickPasses will only be issued for windows that are farther away than the current regular line wait time. This means that everyone in line at the time someone obtains a QP from the kiosk will have already ridden by the time the issued QP window begins. Thus, all QP holders arrived before the regular line riders for the designated QP window: the QP riders were there first, but they were allowed to leave and return. Thus, in an ordinal list of the riders for a window, QP riders must go first.

Analyzing the table, we can draw two conclusions that the previous graphs have suggested. First, for every person in line, the wait time in line under a QuickPass system is less than or equal to the wait time without the QP. Thus, we can state that a QP system, regardless of when the QP holders arrive within the window, is Pareto optimal to not having a QP system. Second, comparing the wait times for the burst versus the even distribution of QP arrival, we see that there is a tradeoff. While the aggregate wait times differ from each other little, who benefits most from the reduction of aggregate wait does vary.

[5] This QuickPass model is in no way affiliated with Gap Inc.

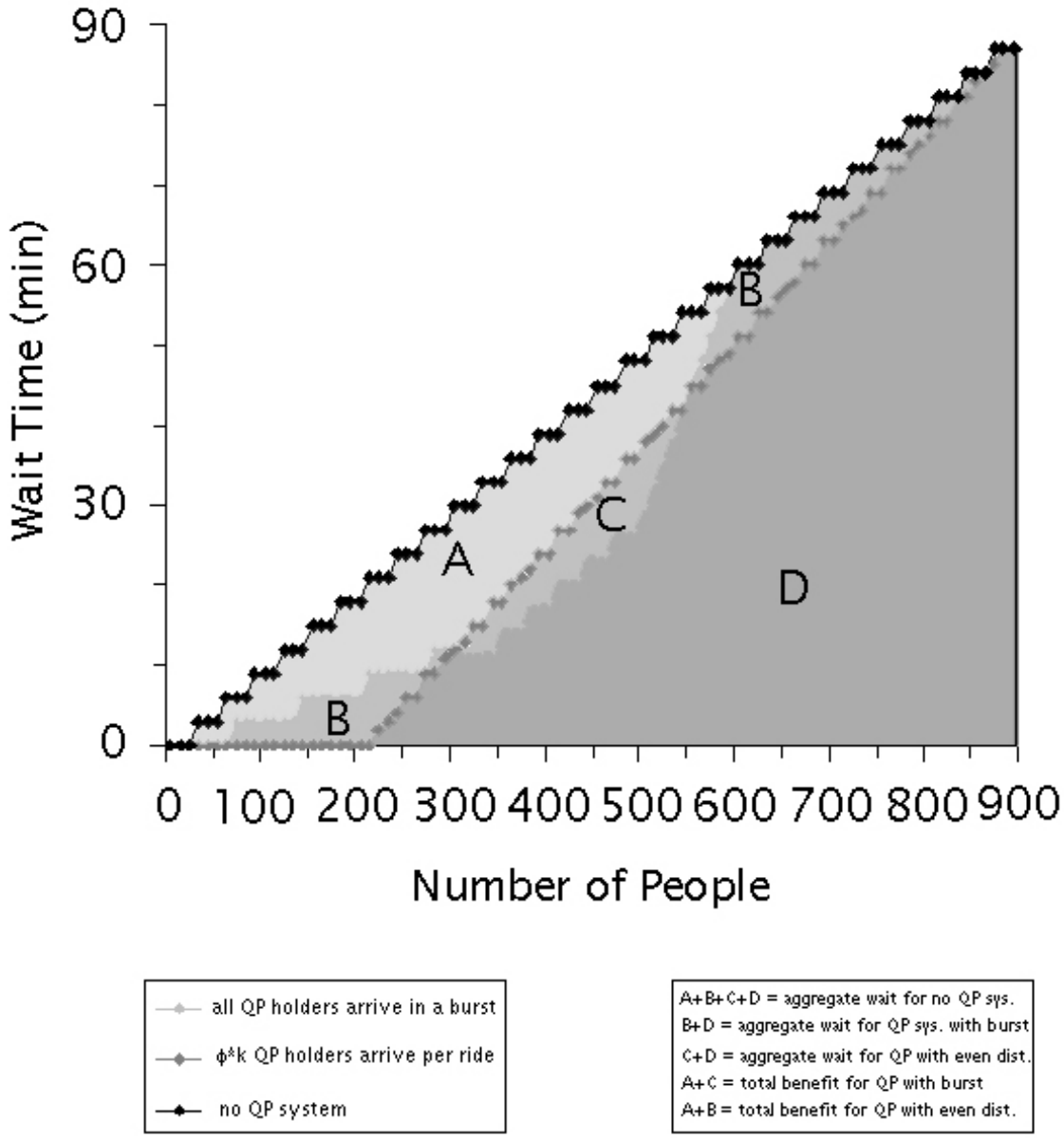

Figure 2: Wait Times & Aggregate Wait for a Burst versus No Burst

Thus, a burst primarily affects the distribution of QP benefits between the separate line members. QuickPass holders are most benefited with an arrival rate dispersed at ϕ_{avg}, though, as the graph shows, nearly all 900 riders experience some benefit in either case, and none are worse off than they would be if there was no QP system.

Furthermore, since QP holders have the lowest possible wait time when they arrive evenly dispersed throughout the window, we propose placing a suggested return time on all QP tickets issued. This suggested time will be within the QP window, and it is not intended as an enforceable return time. Thus, it should not increase stress on the park attendee to return at a specific moment. However, the QP holder should be advised that if he, and everyone else with a QP, returns at or near their suggested time, their already greatly reduced wait time will be even lower, or perhaps nearly zero.

Nth Person In Line	Even QP Arrival	QP burst in middle	No QP System	Nth Person In Line	Even QP Arrival	QP burst in middle	No QP System	Nth Person In Line	Even QP Arrival	QP burst in middle	No QP System
10	0	0	0	310	24	18	30	610	51	60	60
20	0	0	0	320	27	21	30	620	51	60	60
30	0	3	0	330	27	21	30	630	54	60	60
40	0	3	3	340	29	21	33	640	54	63	63
50	0	6	3	350	30	24	33	650	56	63	63
60	0	6	3	360	0	0	33	660	57	63	63
70	0	9	6	370	0	0	36	670	58	66	66
80	0	9	6	380	0	3	36	680	60	66	66
90	0	12	6	390	0	3	36	690	60	66	66
100	0	12	9	400	0	6	39	700	63	69	69
110	0	0	9	410	0	6	39	710	63	69	69
120	0	0	9	420	0	9	39	720	65	69	69
130	2	0	12	430	0	9	42	730	66	72	72
140	3	3	12	440	0	12	42	740	67	72	72
150	4	3	12	450	0	12	42	750	69	72	72
160	6	3	15	460	31	24	45	760	69	75	75
170	6	6	15	470	33	24	45	770	72	75	75
180	9	6	15	480	33	27	45	780	72	75	75
190	9	6	18	490	36	27	48	790	74	78	78
200	11	9	18	500	36	27	48	800	75	78	78
210	12	9	18	510	38	30	48	810	76	78	78
220	13	9	21	520	39	33	51	820	78	81	81
230	15	12	21	530	40	36	51	830	78	81	81
240	15	12	21	540	42	39	51	840	81	81	81
250	18	12	24	550	42	42	54	850	81	84	84
260	18	15	24	560	45	45	54	860	83	84	84
270	20	15	24	570	45	48	54	870	84	84	84
280	21	15	27	580	47	51	57	880	85	87	87
290	22	18	27	590	48	54	57	890	87	87	87
300	24	18	27	600	49	57	57	900	87	87	87

6.3 Queue Simulation [6]

We have developed a program that simulates queues for both the regular line and the Quick Pass (QP) line over time, taking into account additional QP holders that enter the line. The program currently obtains initial parameters from the user, but in a real-life situation, some of these parameters would be coded in as constants, while others would be obtained from the environment. The first type of parameter includes ϕ_{max}, ride capacity, length of window, and maximum allowable wait time for QP holders. The latter type of parameter includes the current number of people in the regular line, the current number of people in the QP line, and then, later on, how many QP holders arrive before each ride departs.

This particular simulation looks at the time of 2 window lengths, for situations in which huge bursts of QP holders arrived at the end of one window and the beginning of the next, as mentioned above.[7] The rest of the simulation lasts as long as there are still people in line. We first look at the maximum number of QP holders that can get on the next ride (we arrive at this value by multiplying $\phi * k$) – if it is less than the total number of QP holders in line, then we will let in the maximum number of QP holders, and fill up the rest of the ride with folks from the regular line. Otherwise, we will merely let in all the QP holders in line, and then fill up the rest with people from the regular line. After each ride, the system determines (either manually or from the environment) how many QP holders have entered the line since the last ride.

We have included in the appendix a sample output for the program using the following parameters: 400 folks in the regular line; 50 people in the QP line; ϕ_{max} is 2/3; 30 people per ride; 3 minutes per ride; 45 minute window; and maximum wait of 15 minutes for QP holders. As you can see in the appendix, varying amounts of QP holders have arrived at different points during the simulation. These are merely arbitrary; as long as the total number of QP holders that arrive does not exceed the total number of QP's issued, the simulation runs smoothly.

[6] See Appendix A for Computer Code & Sample Output
[7] See discussion of bursts in Section 7.2.

7. The Model, Part II

7.1 Derivation of λ

To improve the estimate of wait time for the regular line, λ, we propose the following procedure. To start, let us consider how the day has been divided into windows for QuickPass riders. In each window, N_{tot} people will ride the ride: N_{QP} from the QuickPass line and N_{reg} from the regular line. Let us simplify things for the moment and assume that the maximum number of QuickPasses in each window has been issued – which, in most cases, is a reasonable assumption (given that the rides for which we are issuing QuickPasses are in high demand). This means that N_{QP} and N_{reg} will be the same for every window, with N_{reg} equaling:

$$N_{reg} = \frac{k}{r}\tau_W - \frac{1}{\upsilon}\frac{\phi_{\max} k}{r}\tau_\ell, \qquad \textit{(eq. 10)}$$

$$\text{where } \upsilon = \frac{2\tau_\ell - \tau_g}{\tau_\ell}. \qquad \textit{(eq. 7)}$$

For now, we will set $\tau_g = \tau_\ell$ so that there is no potential overlap of QP holders from consecutive windows. Since $\phi_{\max}$ must be less than 1, as we have previously established, we know that on every ride there will be a rider from the regular line. Because of this, the last person in the regular line who will ride in a given window, the N_{reg} th person, will ride on the last ride of the window, and his wait time during that window will be the length of the window, τ_W.

If we view λ in relation to the division of the day into windows, we can represent λ as the sum of three elements: (a) the partial window the current time is in, (b) subsequent full windows that λ spans, and (c) a partial window at the end that λ does not fully span. λ, of course, is dependent on the total number of people in the regular line, N_{line}, among other things. We will now derive equations for each of the three parts of λ.

By simply subtracting the current time from the time the current window ends, we obtain an exact value for (a). The equation for this component of λ, however, does not look that simple:

$$\lambda_a = \tau_W \left\lceil \frac{\tau_{cur} - \tau_{open}}{\tau_W} \right\rceil - (\tau_{cur} - \tau_{open}) \qquad \textit{(eq. 13)}$$

λ_a is derived from three data inputs: (i) the time of day, given by an internal clock in the kiosk and converted to minutes; (ii) time of beginning of first window, i.e. the time of park opening, converted to minutes; and (iii) the window length in minutes. Inputs (ii) and (iii) are predetermined constants. The equation for λ_a takes the amount of time since the park has opened, $\tau_{cur} - \tau_{open}$, and divides it by τ_W to yield the number of windows that have passed since the park has opened. Taking the ceiling of this quotient tells us what the next window will be, i.e. if the quotient is 5.2, the next window will be the sixth. Multiplying this integer by τ_W tells us the time (expressed as minutes since the park's opening) that the next window will begin, which is the same as when the current window ends. Subtracting the

amount of time since the park has opened, we are left with λ_a, which is the amount of time from the current time until the end of the current window.

We have established that N_{reg} people from the regular line will ride in each window. If we know how many people from the regular line have already ridden the ride in this window, $N_{reg_{passed}}$, we can easily calculate how many more will ride by the end of the current window. Subtracting this value from N_{line}, we are left with the number of people in the line who will ride starting in the next window. This number is given by $N_{line} - (N_{reg} - N_{reg_{passed}})$. We will, of course, need a way to track N_{line} and $N_{reg_{passed}}$. This can be accomplished by a system of people counters wired into the kiosk system, either with turnstiles that people must pass through both to get in line and to board the ride, or with similarly wired counters manually operated by ride attendants. Presumably, a counting system is already in place in amusement parks using a QuickPass-like system, so the data input that our kiosk régime requires might not necessitate an increased expenditure to implement.

Taking $N_{line} - (N_{reg} - N_{reg_{passed}})$, we can now calculate part (b) of λ, which is the amount of time dictated by the number of full windows the line spans. λ_b is given by the following equation:

$$\lambda_b = \tau_W \left\lfloor \frac{N_{line} - (N_{reg} - N_{reg_{passed}})}{N_{reg}} \right\rfloor \qquad \textit{(eq. 14)}$$

Dividing $N_{line} - (N_{reg} - N_{reg_{passed}})$ by N_{reg} and taking the floor of the quotient, we get the number of full windows that the remaining segment of the line spans. Multiplying by τ_W gives us the amount of time it will take this many windows to pass. We can do this because we know that the N_{reg} th person will be on the last ride of the window (his wait time will merely be τ_W). Note that if the length of the line after λ_a is less than N_{reg}, the floor function will equal zero, and $\lambda_b = 0$. We can now add λ_a and λ_b:

$$\lambda_{a+b} \equiv \lambda_a + \lambda_b = \tau_W \left(\left\lceil \frac{\tau_{cur} - \tau_{open}}{\tau_W} \right\rceil + \left\lfloor \frac{N_{line} - (N_{reg} - N_{reg_{passed}})}{N_{reg}} \right\rfloor \right) - (\tau_{cur} - \tau_{open}), \quad \textit{(eq. 15)}$$

where $\lambda_{a+b} = f(N_{line}, N_{reg_{passed}}, \tau_{cur})$.

Now let us approach part (c) of λ, which is the amount of time it will take the last part of the regular line, N_{left}, which includes only those people who will not ride in λ_{a+b}. This number of people is given by $N_{left} = \lfloor N_{line} - (N_{reg} - N_{reg_{passed}}) \rfloor \bmod N_{reg}$, which is the remainder of the quotient used in calculating λ_b. Although we know that N_{reg} people will ride in τ_W, we cannot be certain, however, how long it will take N_{left} (where $N_{left} < N_{reg}$) people to ride because we cannot predict when QP holders will arrive. We can, however, predict a maximum and minimum bound for the wait time, as well as project an average value.

To calculate the range, recall the formula for N_{reg} in a window that does not overlap (i.e. $\upsilon = 1$):

$$N_{reg} = \frac{k}{r}\tau_W - \frac{\phi_{\max} k}{r}\tau_\ell \qquad \textit{(eq. 10)}$$

To analyze this formula in a more intuitive way, let's add and subtract $\frac{k}{r}\tau_\ell$ on the right-hand side as follows:

$$N_{reg} = \frac{k}{r}\tau_\ell - \frac{\phi_{\max} k}{r}\tau_\ell + \frac{k}{r}\tau_W - \frac{k}{r}\tau_\ell$$

Now, collect terms in the following way:

$$N_{reg} = \frac{(1-\phi_{\max})k}{r}\tau_\ell + \frac{k}{r}(\tau_W - \tau_\ell)$$

This formula defines N_{reg} in terms of the extreme case in which all QP holders for the window arrive in one burst. For a duration of τ_ℓ, the rate at which regular line members can ride is discounted by $(1-\phi_{\max})$, and for the rest of the window, $\tau_W - \tau_\ell$, people from the regular line can ride at the full capacity rate $\frac{k}{r}$. Thus, we can view N_{reg} as the sum of two components, one in which people from the regular line board the ride at the slowest possible rate, and another in which they ride at the fastest rate. Let us, then, break N_{reg} down into these two components, which we will label N_{slow} and N_{fast}:

$$N_{slow} = \frac{(1-\phi_{\max})k}{r}\tau_\ell;\ N_{fast} = \frac{k}{r}(\tau_W - \tau_\ell). \qquad \textit{(eq. 16, 17)}$$

Using N_{slow}, N_{fast}, and their corresponding rates, we can project a maximum and minimum value for λ_c, and thus for λ.

For any given person in the regular line, his wait time during the window in which he boards the ride (the last partial window) is longest when all of the QP holders arrive at the beginning of the window. Because the QP holders must ride within τ_ℓ, they take seats that could otherwise be filled with people from the regular line. The regular line moves at a rate discounted by $(1-\phi_{\max})$ until all of the QP holders ride, then it moves at the full, fast rate. That is, N_{slow} goes before N_{fast}.

Expressing this fact, the following system yields $\lambda_{\max}$:

$$\lambda_{\max} = \lambda_{a+b} + p\left(\frac{r}{(1-\phi_{\max})k}N_{left}\right) + q\left(\phi_{\max}\tau_\ell + \frac{r}{k}N_{left}\right) \qquad \textit{(eq. 18)}$$

$$\textit{if } N_{left} \le N_{slow}, p = 1 \textit{ and } q = 0$$

$$\textit{if } N_{left} > N_{slow}, p = 0 \textit{ and } q = 1$$

This means that if the number of people left in this window is less than or equal to the number of people that can be processed at the slow rate, the amount of time to add to λ_{a+b} is the amount of time it takes to accommodate N_{left} people at the slow rate. If $N_{left} = N_{slow}$, this amount of time is τ_ℓ. On the other hand, if $N_{left} > N_{slow}$, the people left in line after

τ_ℓ, $N_{left} - N_{slow}$, will ride at the rate corresponding to the full capacity of the ride. This expression simplifies to the terms in $\lambda_{\max}$ that are multiplied by the dummy variable *q*.

The system for the minimum bound for λ, $\lambda_{\min}$, is analogous:

$$\lambda_{\min} = \lambda_{a+b} + p\left(\frac{r}{k}N_{left}\right) + q\left(\tau_W - \tau_\ell + \frac{r}{(1-\phi_{\max})k}N_{left} - \frac{\tau_W - \tau_\ell}{1-\phi_{\max}}\right) \qquad \textit{(eq. 19)}$$

$$\textit{if } N_{left} \le N_{fast}, p = 1 \textit{ and } q = 0$$

$$\textit{if } N_{left} > N_{fast}, p = 0 \textit{ and } q = 1$$

We also can project an average value for λ, λ_{avg}. This is the case in which QP holders return at their suggested return time, thus evenly spreading out the QP riders over the entire window. The following equation captures this case:

$$\lambda_{avg} = \lambda_{a+b} + \tau_W \frac{N_{left}}{N_{reg}} \qquad \textit{(eq. 20)}$$

Note that, despite the lengthy equations, $\lambda_{\min}$, λ_{avg}, and $\lambda_{\max}$ are all functions of just three variables τ_{cur}, N_{line}, and $N_{reg_{passed}}$. The rest of the terms are constants with predetermined values.

7.2 The Ψ Factor

Heretofore, we have assumed that two people who have each been issued a QuickPass *for* different times of the day *at* different times are equally likely to use that QuickPass. However, it is more likely the case that as the return time gets farther away from the current time *and* as the return time becomes later in the day, the QuickPass is less likely to be used. Unfortunately, we do not have accurate data sets at this juncture with which we could analyze and determine exactly what this depreciation factor is. We do believe that some sort of negative correlation exists between, on the one hand, the time of day(δ_{day}) and the number of hours until the return time(h_{ret}) and, on the other hand, the percentage of people holding QuickPasses that won't show up for a given window. We will call this the ψ factor, with constants α and β:

$$\psi = \alpha h_{ret} + \beta\delta_{day} + 1 \qquad \textit{(eq. 21)}$$

The first part of the formula, without the 1, yields an estimated percentage value of QuickPasses that will not show up. Thus, we add 1 to ψ in order to multiply ψ by the number of QuickPasses and generate the number of QuickPasses we should put out. Here, we simply multiply the two variables by constants. We are assuming that the variables are dependent on each other and ψ is veritably equally dependent on both.

However, it is possible that ψ is more dependent on one variable or the other. The following formula proposes a ψ that it is heavily dependent on the time of day, with a new constant of γ:

$$\psi = \alpha h_{ret} + \beta e^{\gamma\delta_{day}} + 1$$

Of course, there are countless other variations on this formula. In order to truly understand the essence of ψ, one must become one with the data sets. In doing so, it would become clearer as to how to arrive at a more accurate ψ. The proper technique we would follow upon discovering this elusive data, which our amusement park has so conveniently kept from us, would be to fit a regression equation to the data, which would lead us to the secret of the ψ factor.

Now that we understand λ and ψ, we can explore a system that simulates a real-life amusement park QuickPass system.

7.3 QuickPass Kiosk System[8]

7.3.1 Overview of Simulation

We have developed a simulation of the QuickPass Kiosk System (QPKS) that incorporates all the factors we have discussed up to this point. This particular system will ask the user to input certain variables that the actual implementation of the system would be able to obtain from its environment (e.g., Current Time, Current Number of People in Line, etc.).

The QPKS starts by calculating the number of QuickPasses (QP's) that it should distribute in any given window:

$$N_{QP} = \left\lceil \psi \phi k \frac{\tau_W}{r} \right\rceil$$

Here, we multiply the percent of QP's per ride times the number of seats per ride in order to find out how many people with QP's would be on each ride. This is multiplied by the depreciation value, ψ, in order to account for QP holders not showing up during their allotted window due to it being either so late in the day or so many hours away (the derivation of ψ can be found in Section 8.2). The length of the window divided by the duration of the ride will tell us how many rides there will be during the course of the window. When multiplied together, we get the total number of QP's we can dispense during the window.

Next, the number of windows is calculated by simply subtracting the opening time from the closing time, and dividing by the length of the window. An array is created to store how many QP's have already been distributed for any given window. Of course, all of these are initially set to zero. The rest of the program is a large 'while' loop that will continue until the park closes (i.e., until the current time is greater than the closing time; thus, never allowing a QP to be issued after we close).

At the start of each iteration, the system will obtain the current time, the current number of people in the regular line, and the number of people who have gotten on the ride during this window (these are variables that are entered manually in this particular simulation, but would be obtained automatically by the actual kiosks).

The number of people from the regular line that will ride during this window is derived from the length of the window, the capacity of the ride, the length of the ride, and how many QP's have been distributed for the current window:

$$\frac{\tau_W}{r} k - N_{QP}$$

[8] See Appendix B for Computer Code & Sample Output

Once again, we are dividing the length of the window by the length of the ride to get the number of rides per window. By multiplying by the capacity of the ride, we arrive at the total number of people that will ride during this window. Finally we subtract the number of QP's that have been given out for this window to obtain the number of people from the regular line that will ride during this window.

The QPKS evaluates the estimated wait time for the regular line, λ, as a summation of the wait time for the remainder of the window we are currently in, the subsequent windows that λ spans, and the partial window at the end that λ does not fully span. [An in-depth discussion of the derivation of λ can be found in Section 8.1.] If λ is less than the lower bound wait time for doling out QP's, then we will refuse to issue a QP for the time being.

Otherwise, we arrive at a loop that will take us from the first window that's at least λ minutes away to the last window before closing time. Within the loop, we first check to see whether we've already distributed the maximum number of QP's for the *i*th window. If we still have some left, the QPKS tells the user that there are indeed QP's available for that window. Additionally, it will tell the user how many QP's are left. At first consideration, we favored the idea of only allowing the system administrator of the QPKS to obtain this data; however, upon further reflection, we realized that it might be important to a user: if there are only 2 QP's left for a given window, and I'm in a party of 4, then we would rather stay together and wait a little bit longer, than split up and have half of us go now and half go later. This thoughtful consideration by the Amusement Park on behalf of the consumer lacks expensive cost, yet its benefit could be immense in terms of the pleasure and relaxation of the consumer.

After displaying all available windows, the system asks whether the user would like to obtain a QP at this moment; if the answer is affirmative, then it asks which window to obtain a QP for and properly alters the array of how many QP's have been distributed already. An extension of the QPKS that is not included here, due to the unfortunate constraints of the space time continuum, would be to issue an exact return time within the window that is a function of how many QP's have been issued for that period of time. It can be explained to the guest that if he was to arrive at that time, they would have minimal wait time.

And we return back to where we started (to the 'while' loop that goes until closing time).

7.3.2 Discussion of Sample Data

Also in Appendix B are sample outputs for three different situations. This will give the reader a very rough idea of what the interface would be like if this was an actual QPKS. Information that is entered manually but would normally be obtained from the environment is noted. The QPKS gives the guest an idea of how long the wait would be if he/she waited in the regular line. While the first QP window offered will always be further away than if the guest chose to stand in the regular line, the guest might rather desire to spend his/her time in a different way.[9]

The first set of sample output has the following parameters: the current time is 9:35am; 400 people are currently in the regular line; and 40 people from the regular line have already ridden during this window. The QPKS computes the wait time to be 49->59

[9] Please see Section 7.1 for an In-Depth Discussion on the Revenue Benefits of the QP system

minutes, with an estimated wait time of 51 minutes. There are 14 available windows, starting with 11:00-11:30am and ending with 8:45-9:15pm.

The second set of sample output has the following parameters: the current time is 12:25pm; 800 people are currently in the regular line; and 160 people from the regular line have already ridden during this window. The QPKS computes the wait time to be 76->86 minutes, with an estimated wait time of 83 minutes. There are 10 available windows, starting with 2:00-2:30pm and ending with 8:45-9:15pm.

The third set of sample output has the following parameters: the current time is 3:45pm; 2000 people are currently in the regular line; and 100 people from the regular line have already ridden during this window. The QPKS computes the wait time to be 255->255 minutes, with an estimated wait time of 255 minutes. There are 2 available windows, starting with 8:00-8:30pm and ending with 8:45-9:15pm. Note that the maximum and minimum wait times generated are the same. This is because, as incorporated in the calculation for λ, every N_{reg} th person rides in τ_W. The maximum and minimum values converge as N_{left} approaches N_{reg}, and they equal each other at N_{reg}. Thus, the number of people in the regular line in this example turns out to be the exact number that will finish a window, and the last person in line will ride at the end of the window.

8. Extensions

8.1 λ Calculation

Our calculation for λ relied on several assumptions. Here, let us relax them and discuss how the procedure will shift. First, consider the case when $N_{line} - (N_{reg} - N_{reg_{passed}})$ is negative. This means that λ will be less than rest of current window. In this case, λ_{a+b} should be set equal to 0, and the procedure for calculating λ should skip to part (c).

Another case that will alter our calculation of λ is when there is potential overlap between windows, i.e. $\tau_g < \tau_\ell$. As our graphs above have shown, instead of every N_{reg} th rider passing through in τ_W, every $2N_{reg}$ th person rides in $2\tau_W$. In case of a burst of QP riders where the windows meet, it takes until the end of the second window to even out, whereas with no overlap, the line length and window duration line up at the end of each window. Thus, we will amend λ_b as follows:

$$\lambda_b = 2\tau_W \left\lfloor \frac{N_{line} - (N_{reg} - N_{reg_{passed}})}{2N_{reg}} \right\rfloor$$

Likewise, the calculation of λ_c will change slightly. The method for determining the maximum, minimum, and average values is analogous to the method used above. Since we have determined that the best case is when $\tau_g = \tau_\ell$, we will omit the rest of the full derivation.

Similarly, one final case that would alter the projection of λ is if we relax the assumption that the maximum number of QuickPasses from each period (given the set values for the constants) will be issued, although we do not expect this to be a regular occurrence. This new calculation will require no new data inputs, for the kiosk program already records the number of QPs issued. If we give N_{reg} a subscript parameter of θ such that each window θ has a specific N_{reg}, we can compute λ, albeit with a much more complex computation. A final version of the kiosk program would want to include this procedure to ensure optimal wait time measuring in all circumstances, even unlikely ones such as this.

8.2 Integration of the QuickPass Kiosk System

The current QuickPass system involves kiosks that are ride-specific and are only available near that particular ride. This system is inefficient in that someone on one side of the park could be interested in riding a ride on the other side of the park. If he could tell at this moment that there is a long wait, then a QuickPass could be obtained, the guest could meander at will across the park, and show up to the ride during their allotted window, thus minimizing total wait. He would not have to walk all the way across the park to do this, thereby saving him lots of wasted time and energy.

Our idea is to place kiosks throughout the park, with several located near the entrances. The kiosks would be plugged into an integrated system that would be tapped into every ride and would provide real-time data on current wait times, etc. A central database

would be maintained that would contain information on which guests (identified by their daily park entrance ticket) currently have QuickPasses out, so as to ensure that no one obtains multiple passes.

One possible extension of this would be to allow guests (in particular, those who like a schedule to follow) to obtain multiple QuickPasses for different rides throughout the day. Possible features of this system would include the option to create a recommended itinerary by ordering guests' rides into a logical sequence so that they wouldn't have to zigzag all over the park. The demand for this already exists, as evidenced by the scheduling program called RideMax™ for Disneyland®.[10] The essence behind RideMax™ is that it allows you to create a detailed schedule for your day, given your arrival time, your departure time, and which rides you want to ride. While RideMax™ relies on historical wait time data based on time of year, this amusement park could use our model in order to determine more accurate wait times based on the dependent factors and real-time data. It is our recommendation that the amusement park integrate a system such as this into its QuickPass Kiosk System.

The integration will require some sort of computer network to connect all the kiosks together. The obvious first choice would be to use the IEEE 802.3 standard. However, if this implementation is too costly for the amusement park (this is dependent partially on what equipment is currently in use at the park), perhaps it might be more effective to use IEEE 802.16, broadband wireless. This will ensure high speed connections between the kiosks and the central database, thus allowing park guests using the kiosk to get their Quick Passes easily and quickly. Additionally, it will circumvent the need to lay down miles of cable. The park would have to install a single broadband wireless tower, as well as wireless devices in all of its kiosks. A further advantage is future flexibility, for a wireless network facilitates the addition of kiosks down the road. [As an aside, the amusement park could also offer wireless internet to its guests through this system.]

8.3 Artificial Intelligence

No 21st Century computer system would be complete without the latest and hottest technology, artificial intelligence, or simply, A.I.. While the QuickPass Kiosk System won't exactly be dodging bullets quite like Agent Smith from the hit movie, *The Matrix*, a unique extension to this system would be to add the capabilities of a fairly simple "rational agent." One element of this is that the QPKS would determine error between its calculations and actual data. For example, when it calculates the estimated wait time for the folks standing in the regular line, it could store that information and compare it to the actual wait time for those people. Over time, the system would integrate its relative error into the equations. Of course, one wouldn't want to rely heavily on one day's worth of data. The data should be accumulated over a fixed period of time, and then the error would be integrated into the equations.

This field of A.I. is called learning. As the rational agent perceives nuances in its environment, it is able to improve itself over time. Of course, if the QPKS were to have a plethora of data a priori, then the relative learning from the environment would be much less (although it could still have impact on the calculations).

[10] RideMax Software

9. Conclusions: Strengths and Weaknesses

The principal strength of our model lies in the idea that the maximum allowable time, as set by the amusement park, that a QP holder will wait in line will never be exceeded. Additionally, no QP's will be given out for a return time sooner than a previous distributed QP. Thus, we have satisfied the problem given to us. However, we have taken this further. We have included the ability to have multiple QP's. This will allow the guest to plan their day in advance, or perhaps even be issued a rough itinerary for the day!

In the beginning of this paper we quoted Julie's complaints about Disneyland's® FastPass system. We then briefly discussed how we would go about dealing with these complaints, and incorporated them into our model throughout the paper. A concise list of how we turned those complaints around follows: more accurate estimates for wait times, QuickPass holders will never fill a ride (thus, the regular line is always moving), QuickPass users will have the ability to choose from all available windows, all men ARE created equal, and the ability to ride back-to-back on a single ride is allowed, in addition to those strengths listed in the paragraph above.

The general conclusions derived from our model are independent of assumed values in the examples we used. The park should be able to enter in whatever it desires. To follow this line of thought, special events won't affect our system. The QP system is in place to redistribute peaks amongst a longer period during the day. Special events will most likely just increase the level of those daily peaks. Also, if a ride breaks down, it is likely that the integrity of the QP system will be maintained. We take into account the maximum wait time for QP holders; only if a large burst of them arrive at the same time will we need a buffer. Otherwise, we could have some time to fix the broken ride.

Finally, we believe that the cost of implementing this system is worth it, considering the high returns on people's enjoyment & relaxation, not to mention increased revenue!

There are some weaknesses in our model. As Obi-Wan Kenobi says, "The Force can have a strong influence on a weak mind."[11] However, we will expound on our weaknesses; upon further consideration, it would be possible to take care of these, pending additional time. The model presented in this paper does not model flow of people into the park or into a given ride. Therefore, it won't project wait times for a certain point in the day. It merely takes in given parameters and outputs estimated wait times. In other words, the model works at the margin. If the extension regarding Artificial Intelligence was put into action, then this weakness could be taken care of easily.[12]

We also don't take into account that the length of the line is going to affect the decision of whether or not to take a QP. If there is a long line at the kiosk, it will probably also affect the person's decision. This brings up the question of whether there is an optimal number of QP kiosks. We suspect there is, but we leave it as an exercise for the reader.

In general, using accurate data will yield good results. This is especially true when dealing with the A.I. extension of our model.

We hope you enjoyed your ride, please mind the gap as you exit to the right…

---

[11] Star Wars, 1977.
[12] Please refer to Section 9.3.

10. Non-Technical Summary

QuickPass, Inc.

To: Amusement Park Executives

From: QuickPass, Inc.

CC: MCM Judges

Date: May 31, 2004

Re: The QuickPass System™

Greetings and Salutations! We have recently concluded an exhaustive mathematical study of various QuickPass implementations per your request that we received approximately 96 hours ago. We have theorized and philosophized about the in's and the out's, the strengths and the weaknesses, and the good, the bad, and the ugly of these systems. We began our approach by analyzing the problem as stated in the 2004 COMAP Mathematical Contest in Modeling.

Immediately, we can ensure that under our system, people holding QuickPasses will never wait in line more than the maximum allowable wait time, which you can select [alternately, with more time and additional data specific to your park, our consultants would be happy to recommend this parameter for you]. With our system, no one will be able to get a QuickPass for a time earlier than one already handed out. A person standing in the regular line will always ride before a person who obtains a QuickPass after the first person gets in line.

During the course of our investigation, we discovered a few comments and complaints from a recent guest at an amusement park that will remain nameless, in order to protect the guilty (the amusement park in question employs something similar to our QuickPass system, yet unfortunately inferior). To summarize her comments:

- Estimated time waiting in the regular line was frequently off.
- The regular line often took longer than expected, due to a sudden burst of "QuickPass" holders.
- The "QuickPass" system allowed her to ride a ride twice in a row.

Rest assured, we have addressed the above comments, as well as many others (see attachment). We have created a system in which more accurate estimates for wait times are available. These are updated in real-time, depending on various considerations, such as the current number of people standing in the regular line. Also, QuickPass holders will

never fill a ride, thus guaranteeing that the regular line will always be moving. We have set a maximum number of QuickPass holders that can get on any given ride [once again, this number can be set by you, or our consultants would be happy to recommend one for you, pending further time and more information about your particular park]. While the current systems in use today only allow the guest to have one QuickPass at a time, we have found that many of our visitors would rather have the ability to have multiple QuickPasses. Our system incorporates this ability, should you desire it. Also, several people have taken advantage of a hole in which one can ride back-to-back on a single ride. This can occur, since when a person obtains a QuickPass for a given ride, the current wait time in the regular line is always less than the time until the window of time for which the QuickPass has been issued. We have maintained this feature of the QuickPass system, although it, too, could be removed at your particular park at little added inconvenience.

Finally, we would like to state that we agree with the Declaration of Independence: All Men *are* Created Equal. A few visitors have complained that their line moves slower while the QuickPass holders enter their respective line and ride very quickly. However, we have made certain that overall, the not one person is worse off than they would be if the system didn't exist. Furthermore, the majority of people will benefit from this system. The regular lines will be shorter, and the people choosing to obtain a QuickPass can use their time in ways other than waiting in line. Some examples of what they might do are ride another ride, buy food or drink, purchase merchandise, or play one of the many games that are popular at amusement parks. Clearly, one does not need an MBA to see that this QuickPass system can be financially beneficial to your amusement park. Using actual data acquired from an amusement park along with our own sample data, we calculated that using our QuickPass system, we could generate approximately $192 million a year in added revenue for this particular park, exclusively from the time saved waiting in line.

Thus, not only will your guests be happier and more relaxed as they visit your park, but you will also see your profits surge because of the QuickPass system! This should more than cover the cost for actually implementing our system.

We hope that you will contact us as soon as possible to discuss further the feasibility of using our QuickPass system in your amusement park. We look forward to meeting with you.

Sincerely,

The Consultants
QuickPass, Inc.

Attachment: Mathematical Investigation of the QuickPass System

11. Appendices

11.1 Appendix A

Queuing: Computer Simulation

(Written in C++, using the g++ Compiler)

```
#include <iostream>
#include <math.h>
using namespace std;

int main(){
        int n; //people in regular line
        int nQP; //people in QP line
        float phi; //max% people to let on from QP line
        int k; //number of people per ride
        int r; //time per ride
        int tWin; //window size
        int tWait; //maximum wait time for QP people

        cout << "Welcome to the Queuing System. Please enter the following"
                << " initial data:\n";
        /*gets initial data from user*/
        cout << "Number of people in the regular line: ";
        cin >> n;

        cout << "Number of people in the Quick Pass line: ";
        cin >> nQP;

        cout << "Maximum Percent of people to let on from QP line: ";
        cin >> phi;

        cout << "Number of people per ride: ";
        cin >> k;

        cout << "Time per ride: ";
        cin >> r;

        cout << "Size of window: ";
        cin >> tWin;

        cout << "Maximum wait time for QP people: ";
        cin >> tWait;

        int tWin2 = 2*tWin;
        int thisRide;
        int newQP;
```

```
        int currTime = 0;
        int nQPRide;

        cout << "At time " << currTime << ", " << nQP << " people in QP line..." << n
                << " people in regular line.\n";

        int i=0;
        while( (tWin2 > 0) && ( (n>0) || (nQP>0) ) ){

                thisRide = k;
                nQPRide = int(ceilf(phi*k));

                if(nQPRide < nQP){
                        nQP = nQP - nQPRide;
                        if(n > (k-nQPRide)){
                                n = n - ( k - nQPRide );
                        }else{
                                n=0;
                        }
                }else{
                        if(n > (k-nQP)){
                                n = n - (k-nQP);
                        }else{
                                n=0;
                        }
                        nQP = 0;
                }
                currTime += r;

                cout << "At time " << currTime << ", " << nQP
                        << " people in QP line..." << n << " people in regular line.\n";

                cout << "How many QP folks have joined in at time "
                        << currTime << "? ";
                cin >> newQP;
                nQP += newQP;


                tWin2 = tWin2 - r;
                i++;
        }
        cout << "<end of simulation>\n";
        return 0;
}
```

SAMPLE OUTPUT for program *[note: italics signifies user input]*

Sample Data: 400 folks in the regular line; 50 people in the QP line; ϕ_{max} is 2/3;
30 people per ride; 3 minutes per ride; 45 minute window;
maximum wait of 15 minutes for QP holders

Welcome to the Queuing System. Please enter the following initial data:
Number of people in the regular line: *400*
Number of people in the Quick Pass line: *50*
Maximum Percent of people to let on from QP line: *.6666666666666666*
Number of people per ride: *30*
Time per ride: *3*
Size of window: *45*
Maximum wait time for QP people: *15*
At time 0, 50 people in QP line...400 people in regular line.
At time 3, 30 people in QP line...390 people in regular line.
How many QP folks have joined in at time 3? *5*
At time 6, 15 people in QP line...380 people in regular line.
How many QP folks have joined in at time 6? *3*
At time 9, 0 people in QP line...368 people in regular line.
How many QP folks have joined in at time 9? *9*
At time 12, 0 people in QP line...347 people in regular line.
How many QP folks have joined in at time 12? *0*
At time 15, 0 people in QP line...317 people in regular line.
How many QP folks have joined in at time 15? *4*
At time 18, 0 people in QP line...291 people in regular line.
How many QP folks have joined in at time 18? *0*
At time 21, 0 people in QP line...261 people in regular line.
How many QP folks have joined in at time 21? *0*
At time 24, 0 people in QP line...231 people in regular line.
How many QP folks have joined in at time 24? *9*
At time 27, 0 people in QP line...210 people in regular line.
How many QP folks have joined in at time 27? *5*
At time 30, 0 people in QP line...185 people in regular line.
How many QP folks have joined in at time 30? *15*
At time 33, 0 people in QP line...170 people in regular line.
How many QP folks have joined in at time 33? *0*
At time 36, 0 people in QP line...140 people in regular line.
How many QP folks have joined in at time 36? *0*
At time 39, 0 people in QP line...110 people in regular line.
How many QP folks have joined in at time 39? *0*
At time 42, 0 people in QP line...80 people in regular line.
How many QP folks have joined in at time 42? *0*
At time 45, 0 people in QP line...50 people in regular line.
How many QP folks have joined in at time 45? *0*
At time 48, 0 people in QP line...20 people in regular line.
How many QP folks have joined in at time 48? *0*
At time 51, 0 people in QP line...0 people in regular line.
How many QP folks have joined in at time 51? *0*

<end of simulation>

11.2 Appendix B

Quick Pass Kiosk System: Computer Simulation

(Written in C++, using the g++ Compiler)

```
#include <iostream>
#include <math.h>
using namespace std;

int minToTime(int tmp);//converts minutes to time (eg, 90 -> 1:30, or 130)
int timeToMin(int tmp);//converts time to minutes (eg, 1:30, or 130 -> 90)

int main(){
	/*variables, and sample values*/
	int lambda,lambdaZ,lambdaAvg,lambdaLow,lambdaHigh; //current wait times
                                                           //for regular line
	int minLambda=30; //minimum wait time to issue QP
	int tCurr=800; //current time
	int tOpen=800; //opening time
	int tClose=2200; //closing time
	float phi=0.22222222; //% QP's per window
	float phiMax=0.66666666; //max % QP's per window
	int k=30; //number of people per ride
	int r=3; //time duration of ride
	int tWait=15; //maximum allowable wait time for QP line
	int nQP; //number of QP's per window
	int tWin=45; //duration of window
	int gap=15; //gap time
	float psi=1; //depreciation factor

	int pplInLine; //current number of people in the regular line
	int pplOnRide; //number of people that have gotten on the ride from the regular
                                // line during this window
	int pplToRide; //number of people from regular line
                                //that will ride during this window
	int nReg; //number of people per window

	cout << "Loading Quick Pass Kiosk...\n"
		<< "\n\nWelcome to the Quick Pass Kiosk!  Please enter some "
		<< information for us.\n"
		<< "\t[All durations should be in minutes (30,56,80,100,...),\n"
		<< "\tall times should be in military form (eg,8:34am -> 834; "
		<< "4:25pm -> 1625)]\n";
/*	cout << "Current/Opening time: ";
```

```
	cin >> tOpen;
	tCurr = tOpen;

	cout << "Closing time: ";
	cin >> tClose;

	cout << "Percent of QP's to issue per window: ";
	cin >> phi;

	cout << "Maximum percent of QP's: ";
	cin >> maxPhi;

	cout << "Total Duration of window (including gap): ";
	cin >> tWin;

	cout << "Length of gap: ";
	cin >> gap;

	cout << "Maximum Allowable Wait Time for QP line: ";
	cin >> tWait;

	cout << "Number of People per Ride: ";
	cin >> k;

	cout << "Time Duration of Ride: ";
	cin >> r;

	cout << "Minimum Wait Time before Issuing QP's: ";
	cin >> minLamda;
*/

	nQP = int(ceilf(float(psi * phi * k * tWin) / r));

	int n = int(floorf(float(timeToMin(tClose-tOpen))/tWin));
	int windows[n];
	for(int j=0;j<n;j++){
		windows[j]=0;
	}

	int tStart,tEnd;
	bool text;

	while(tCurr < tClose){
		//system("clear");
		cout << "\n\nWelcome to the Quick Pass Kiosk!\n";
		cout << "...info to be obtained via the lines/computer...\n";
		cout << "\tCurrent Time: ";
		cin >> tCurr;
```

```
cout << "\tCurrent Number of People in the Regular Line: ";
cin >> pplInLine;
cout << "\tNumber of People who Have Gotten on the Ride\n\tFrom the"
        << " Regular Line during this window: ";
cin >> pplOnRide;

cout << ".....\n";

//nReg = (tWin * k)/r - windows[timeToMin(tCurr - tOpen)/tWin];
nReg = (tWin * k)/r - 100;

lambdaZ = int( (tWin * ceilf(float(timeToMin(tCurr-tOpen))/tWin)
                - timeToMin(tCurr-tOpen))
                + float(((pplInLine - nReg + pplOnRide)/nReg)*tWin) );

pplToRide = (pplInLine - nReg + pplOnRide) % nReg;
lambdaAvg = lambdaZ + (pplToRide * tWin)/nReg;

if( pplToRide <= tWait * ( ((1-phiMax)*k) / r )){
        lambdaHigh = lambdaZ
                        + int(( (r * pplToRide) / (k * (1-phiMax)) ));
}else{
        lambdaHigh = lambdaZ
                        + int((phiMax * tWait) + ( (r * pplToRide) / k));
}

if( pplToRide <= ( float(tWin - tWait) *k)/r){
        lambdaLow = lambdaZ + int( float(pplToRide * r) / k);
}else{
        lambdaLow = lambdaZ + tWin - tWait
                        + int(float( (r*pplToRide)
                        - (k * (tWin-tWait) ))/(k * (1-phiMax)));
}

lambda = lambdaAvg;

cout << "Current wait time for this ride: " << lambdaLow << " to "
        << lambdaHigh << " minutes.\nEstimated wait time: "
        << lambdaAvg << " minutes.\n";

if(lambda < minLambda){
        cout << "We are not currently issuing Quick Passes due to the"
                << " minimal wait time for the regular line.\n";
        continue;
}

text=true;
```

```
                for( int i=int(ceilf(float(timeToMin(tCurr - tOpen) + lambda)/tWin));
                        i < n ; i++ ){   //where n=(tClose-tOpen)/tWin
                        if(windows[i] >= nQP){continue;}

                        tStart = minToTime(timeToMin(tOpen) + (i*tWin));
                        tEnd = minToTime(timeToMin(tStart) + tWin - gap);

                        if(text){
                                cout << "We are currently issuing Quick Passes for the"
                                        << " time periods from..\n";
                                text=false;
                        }
                        cout << '\t' << tStart << " to " << tEnd
                                << "...[" << nQP-windows[i] << " QP's left]\n";
                }
                char yn;
                cout << "Would you like to obtain a quick pass? (y/n): ";
                cin >> yn;
                if(yn == 'y'){
                        int QPstart;
                        cout << "Enter the start time of the time period during which "
                                << "you wish to obtain your quick pass: ";
                        cin >> QPstart;
                        windows[int(ceilf(float(timeToMin(QPstart-tOpen))/tWin))]++;
                }else if(yn == 'n'){
                        cout << "Thanks for using our system!\n\n";
                }else{cerr << "You did not enter y or n! bad boY!\n";}

        }

        return 0;
}


int minToTime(int tmp){//converts minutes to time (eg, 90 -> 1:30, or 130)
        return ((tmp/60)*100) + (tmp%60);
}

int timeToMin(int tmp){//converts time to minutes (eg, 1:30, or 130 -> 90)
        int hrs = tmp/100;
        return (hrs*60) + (tmp-(hrs*100));
}
```

SAMPLE OUTPUT for program *[note: italics signifies user input]*

Sample Data 1: 9:35am; 400 people currently in the regular line;
40 people from the regular line have already ridden during this window

```
$ ./QPkiosk
Loading Quick Pass Kiosk...


Welcome to the Quick Pass Kiosk!  Please enter some information for us.
      [All durations should be in minutes (30,56,80,100,...),
      all times should be in military form (eg,8:34am -> 834; 4:25pm -> 1625)]

Welcome to the Quick Pass Kiosk!
...info to be obtained via the lines/computer...
      Current Time: 935
      Current Number of People in the Regular Line: 400
      Number of People who Have Gotten on the Ride
        From the Regular Line during this window: 40
.........................
Current wait time for this ride: 49 to 59 minutes.
Estimated wait time: 51 minutes.
We are currently issuing Quick Passes for the time periods from..
      1100 to 1130...[100 QP's left]
      1145 to 1215...[100 QP's left]
      1230 to 1300...[100 QP's left]
      1315 to 1345...[100 QP's left]
      1400 to 1430...[100 QP's left]
      1445 to 1515...[100 QP's left]
      1530 to 1600...[100 QP's left]
      1615 to 1645...[100 QP's left]
      1700 to 1730...[100 QP's left]
      1745 to 1815...[100 QP's left]
      1830 to 1900...[100 QP's left]
      1915 to 1945...[100 QP's left]
      2000 to 2030...[100 QP's left]
      2045 to 2115...[100 QP's left]
Would you like to obtain a quick pass? (y/n): y
Enter the start time of the time period during which you wish to obtain your quick pass:
1100
```

SAMPLE OUTPUT for program *[note: italics signifies user input]*

Sample Data 2: 12:25pm; 800 people currently in the regular line;
160 people from the regular line have already ridden during this window

Welcome to the Quick Pass Kiosk!
...info to be obtained via the lines/computer...
Current Time: *1225*
Current Number of People in the Regular Line: *800*
Number of People who Have Gotten on the Ride
From the Regular Line during this window: *160*
.....
Current wait time for this ride: 76 to 86 minutes.
Estimated wait time: 83 minutes.
We are currently issuing Quick Passes for the time periods from..
1400 to 1430...[100 QP's left]
1445 to 1515...[100 QP's left]
1530 to 1600...[100 QP's left]
1615 to 1645...[100 QP's left]
1700 to 1730...[100 QP's left]
1745 to 1815...[100 QP's left]
1830 to 1900...[100 QP's left]
1915 to 1945...[100 QP's left]
2000 to 2030...[100 QP's left]
2045 to 2115...[100 QP's left]
Would you like to obtain a quick pass? (y/n): *y*
Enter the start time of the time period during which you wish to obtain your quick pass:
1445

SAMPLE OUTPUT for program *[note: italics signifies user input]*

Sample Data 3: 3:45pm; 2000 people currently in the regular line;
100 people from the regular line have already ridden during this window

Welcome to the Quick Pass Kiosk!
...info to be obtained via the lines/computer...
Current Time: *1545*
Current Number of People in the Regular Line: *2000*
Number of People who Have Gotten on the Ride
From the Regular Line during this window: *100*
.....
Current wait time for this ride: 255 to 255 minutes.
Estimated wait time: 255 minutes.
We are currently issuing Quick Passes for the time periods from..
2000 to 2030...[100 QP's left]
2045 to 2115...[100 QP's left]
Would you like to obtain a quick pass? (y/n): *y*
Enter the start time of the time period during which you wish to obtain your quick pass:
2045

12. Works Cited

American Beauty. Dir. Sam Mendes. Kevin Spacey, Annette Bening, Thora Birch, Scott Bakula.[13] DVD. DreamWorks. 1999.

"FastPass." Feb 6, 2004. http://allearsnet.com/tp/fastpass.htm. AllEarsNet.

Kirsner, Scott. "Hack the Magic." Feb 6, 2004. http://www.wired.com/wired/archive/6.03/disney_pr.html. Wired Magazine.

Matrix, The. Dirs: Andy & Larry Wachowski. Keanu Reeves, Laurence Fishburne, Carrie-Ann Moss. DVD. Warner Bros. 1999.

RideMax Software. "Disneyland: Spend Less Time in Line." Feb 6, 2004. http://ridemax.com/.

Star Wars. Dir: George Lucas. Mark Hamill, Harrison Ford, Carrie Fisher. DVD. 1977.

[13] You know, the guy from *Quantum Leap*!